الجمهورية الجزائرية الديمقراطية الشعبية

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

وزارة التعليم العالي والبحث العلمي

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université des frères Mentouri Constantine 1

Faculté des Science de la Nature et de la Vie

Département : Biologie Appliquée

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

Intitulé

Apport des métaheuristiques pour la recherche de sous séquences nucléiques ou protéiques

❖ **Présenté et soutenu :**

- **Le :** 08/07/2021
- **Par :** HARCHAOUI Ikram Haifa
  ABID Hasna

❖ **Jury d'évaluation :**

- Président de jury : Dr. BELLIL Ines — MCA. UFM. Constantine 1.
- Encadreur : Dr. DAAS Mohamed Skander — MCB. UFM. Constantine 1.
- Examinateur : Dr. GHERBOUDJ Amira — MCA. UFM. Constantine 1.

Année universitaire : 2020–2021

Remerciements

Nous tenons tout d'abord et avant tout à remercier "**ALLAH**" de tout puissant qui nous a données durant toutes ces années le courage, la volonté, la patience et la foi en nous pour arriver à ce jour.

Nous tenons à exprimer toute notre reconnaissance à notre directeur de mémoire, monsieur **DAAS Mohamed Skander**. Nous la remercions de nous avoir encadrées, orientées, aidées et conseillées.

Aussi nous ne manquons pas de présenter nos vifs remerciements à madame **GHERBOUDJ Amira** d'avoir accepter d'examiner ce travail. Egalement nous n'omettons pas, à remercier madame **BELLIL Inès** d'avoir présidée ce jury.

Nous adressons nos sincères, remerciements à tous les enseignants qui ont assuré notre formation durant les cinq ans et tous les personnes par leur conseils et critiques tout au long nos études.

Nous remercions infiniment nos très **chers parents** lesquels grâce à dieu et à eux nous sommes ici ; ils ont toujours été là pour nous, merci pour tout ce que vous avez fait et entrain de faire pour nous.

Nous remercions nos chers frères et sœurs pour leur encouragement.

Enfin nous remercions nos chers amis pour leur soutien inconditionnel.

Dédicace

Je dédie ce travail,

À ma très **chère mère**,

Pour ton dévouement et tes sacrifices,

Tu es et tu resteras toujours ma source d'inspiration.

Tes prières ont illuminé ma voie et m'ont été d'un grand secours pour atteindre mes buts dans la vie.

Tu as toujours été à mes côtés pour me soutenir, m'encourager et m'épauler. Tu es et tu resteras toujours mon professeur, mon réconfort, et ma meilleure amie.

Tout ce que je suis et ce que j'espère devenir, je le dois à ma très chère mère.

À mon très **cher père**,

Ton soutien a renforcé mon esprit fonceur et ma motivation pour réaliser mes rêves.

Tu es et tu restes toujours mon héros, mon confident, mon bras droit.

A ma sœur **Ilhem**

A mes très chers frères **Sohaib et Adem**

A mes belles **tantes**

A **ma grand-mère** et **mon grand-père**

A mes chères copines **Meriem** et **Sawsen**

A mon cousin **Sami** et ma petite cousine **Ania**

A tous ceux qui m'ont aidé de prêt ou de loin.

Ikram

Dédicace

A mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse, leur soutien et leurs prières tout au long de mes études, et surtout pour leurs encouragements permanents, et leur soutien moral

A mes chers sœurs et BELKISE et DINA pour leur appui et leur encouragement,

A toute ma famille pour leur soutien tout au long de mon parcours universitaire,

A

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible,

Merci d'être toujours là pour moi.

Hasna

Apport des métaheuristique pour la recherche de sous séquences nucléiques ou protéiques

Résumé

La recherche de séquence permet à un utilisateur de rechercher une sous-séquence d'ADN spécifique dans une séquence d'ADN ou une base de données plus grande. Dans ce contexte, les algorithmes de recherche de séquence rapide joueront un rôle important dans l'exploitation des informations contenues dans les données nouvellement séquencées. Les métaheuristiques sont des méthodes approchées et sont des procédures de haut niveau conçues pour résoudre des problèmes d'optimisation. Les métaheuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global, c'est-à-dire l'extremum global d'une fonction, par échantillonnage d'une fonction objectif.L'objectif de notre mémoire c'est d'implémenté un programme avec le langage python basé sur la méthode approcher la particule swarm optimisation (PSO) pour le but de trouvé la sous-séquence la plus longue et le comparé avec une autre méthode exacte la programmation dynamique (DP). Et obtenir un meilleur résultat.

Mots clés : Métaheuristique, optimisation, particule swarm optimisation, recherche de sous séquence.

مساهمة métaheuristiqueفي البحث عن التسلسلات النووية و البروتينية

ملخص

َسًح ٍحث ازرسهسم زهًسرخذو ٍاناحث عٍ حًطـ َىوٌ رَثٍ يحذد الحمًا نٍ ذسهسم أو لاعذج ٍثاُثاَاخ أكثز

نهحًط انُىٌ. نٍ هذا انسُاق، سرهعةَ خىارسيُاخ انثحث انًرسهسم انسزَع دو رًا يه ًًا نٍ اسرغالل

انًعهىياخ انىاردج نٍ انثاُثاَاخ انًرسهسح حذَثًا. عهى انًُرا هٍ غزق ذمزَثُح وهٍ إجزاءاخ عانُح انًسرىي

يصًًح نحم يشاكم انرحسٍُ ذعذ

Métaheuristiquesعًى ًيا خىارسيُاخ عشىائُح ذكزارَح، وازرٍ ذرمذو َحى انحذ األلصً انعانًٍ، أٌ انحذ

األلصً

انعانًٍ زهذانح، عٍ غزَك أخذ عُُح يٍ وظُنح يطىعُح .انهذف يٍ أغزوحرُا هى ذُنُذ ثزَايج نهغخ

ناَثىٌ ثُا ًٍ عهً

غزَمح ذحسٍُ سزب انجسًُُاخ تغزض إَجاد أغىل َرُجح الحمح ويمارَرها يع غزَمح أخزي دلُمح: انثزيجح انذًٍاَُكُح.

وانحصىل عهً َرُجح أفعم.

الكلمات الرئيسية:Métaheuristiques: انرحسٍُ، ذحسٍُ سزب انجسًُُاخ، انثحث ذحد انرسهسم.

Contribution of metaheuristics for the research of nucleic or protein subsequences

Abstract

Sequence search allows a user to search for a specific DNA subsequence within a DNA sequence or a larger database. In this context, fast sequence search algorithms will play an important role in exploiting the information contained in the newly sequenced data.Metaheuristics are approximate methods and are high-level procedures designed to solve optimization problems. Metaheuristics are generally iterative stochastic algorithms, which progress towards a global optimum, that is to say the global extremum of a function, by sampling an objective function.The objective of our thesis is to implement a program with the python language based on the approach the particle swarm optimization (PSO) method for the purpose of finding the longest subsequence and comparing it with another exact method. Dynamic programming (PD). And get a better result.

Keywords: Metaheuristics, optimization, particle swarm optimization, search under sequence.

Table de matière

PARTIE 02 : Partie pratique

Liste d'Abréviations

ADN : Acide désoxyribonucléique.

ARN : Acide ribonucléique.

AA : Acide aminée.

EMBL : European Moleculary Biology Organization.

NCBI : National Center for Biotechnology Information.

DDBJ : Dna Data Bank.

NIG : National Institute of Genetics.

PIR-NBRF : National Biomedical Research Foundation.

MIPS : Martinsried Institute for Protein Sequences.

JIPID : Japan International Protein Information Database.

PSO : Optimization par essaim particulaire.

GA : Algorithme génétique.

ACO : Algorithme des colonies de fourmis.

GBSA : Algorithme de recherche basé sur la galaxie.

CSS : La recherche du système chargé.

SNP : Single nucleotide polymorphism.

Liste des figures

Liste des tableaux

Introduction générale

Introduction générale

Les séquences d'ADN ont été pendant des années une grande préoccupation pour de nombreux travaux de recherche en bioinformatique. La séquence d'ADN est une longue chaîne de caractères spécifiant les nucléotides présentés dans l'ADN. La recherche de séquence permet à un utilisateur de rechercher une sous-séquence d'ADN spécifique dans une séquence d'ADN ou une base de données plus grande. Il sert de bloc vital dans de nombreux domaines tels que la pharmacogénétique, la phylogénétique et la génomique . À mesure que le séquençage des données génomiques devient de plus en plus abordable, la quantité de données de séquence qui doit être traitée augmentera également de manière exponentielle. Dans ce contexte, les algorithmes de recherche de séquence rapide joueront un rôle important dans l'exploitation des informations contenues dans les données nouvellement séquencées.

Dans ce mémoire nous étudions l'apport des métaheuristiques, qui sont des méthodes approchées, à la recherche de la plus longue sous-séquence commune dans une seconde séquence. La première partie de ce mémoire est constituée de deux chapitres. Nous présentons les concepts de base sur les séquences biologiques et les méthodes de recherche de la plus longue sous-séquence commune dans le premier chapitre. Le deuxième chapitre décrit principalement les techniques d'optimisation par les métaheuristiques. La deuxième partie de ce mémoire présentée par le chapitre 3 est consacrée à la présentation de notre travail qui consiste à montrer l'efficacité des méthodes approchées pour le problème de la recherche de la plus longue sous-séquence commune. Ou la métaheuristique PSO sera formulée, implémentée et évaluée pour un tel problème. Dans la dernière partie, une conclusion résume l'ensemble du travail présenté et des perspectives pour de futurs travaux sont suggérées.

CHAPITRE 01 :

Notions sur la bioinformatique et la recherche de sous-séquences

1. Introduction

La bioinformatique est un domaine interdisciplinaire qui développe des méthodes et des outils logiciels pour comprendre les données biologiques, en particulier lorsque les ensembles de données sont vastes et complexes. En tant que domaine scientifique interdisciplinaire, la bioinformatique combine la biologie, l'informatique, l'ingénierie de l'information, les mathématiques et les statistiques pour analyser et interpréter les données biologiques[1]. Les sous-disciplines importantes de la bioinformatique et de la biologie computationnelle comprennent d'une part le développement et la mise en œuvre de programmes informatiques permettant un accès, une gestion et une utilisation efficaces de divers types d'informations, d'autre part, elles comprennent le développement de nouvels algorithmes et de mesures statistiques qui évaluent les relations entre les membres de grands ensembles de données. Bien que les données biologiques soient mémorisées de diverses manières, le texte reste la principale forme de stockage et d'échange d'informations dans les banques de données biologiques. Cela s'applique en biologie moléculaire car les molécules biologiques peuvent souvent être approximées comme des séquences de nucléotides ou d'acides aminés. C'est la raison pour laquelle la recherche de séquences est considérée comme une tâche importante. La recherche de la plus longue sous-séquence (sous-chaine) commune est parmi les opérations les plus utilisées dans les banques de données et consiste à rechercher la plus longue sous-chaîne d'une chaîne qui peut figurer dans un texte. Dans ce chapitre nous présentons quelques concepts de base des séquences nucléiques et protéiques ainsi que les méthodes de recherche les plus utilisées.

2. Pourquoi la comparaison de séquences ?

Dans une séquence d'ADN, ou une molécule d'ADN, il existe quatre bases nucléotidiques : l'adénine, la guanine, la cytosine et la thymine. La connaissance d'une séquence d'ADN et l'analyse de gènes peuvent être utilisées dans plusieurs domaines de recherche en biologie, médecine et agriculture tels que : les diagnostics possibles de maladies ou d'anomalies, la médecine légale, le Pattern Matching, la biotechnologie, etc. Des outils et des méthodes pour accélérer les découvertes et les connaissances dans les sciences liées à la biologie. L'analyse de la séquence d'ADN peut être utilisée pour identifier d'éventuelles erreurs ou anomalies dans une séquence d'ADN (par exemple, par rapport à une séquence normale). Il peut également être utilisé pour prédire la fonction d'un gène particulier et le comparer avec d'autres gènes « similaires » provenant d'organismes identiques ou différents.

Si une nouvelle séquence d'ADN est découverte, sa fonctionnalité est spécifiée en fonction de sa similitude avec des séquences d'ADN connues. Une telle technique est utilisée dans plusieurs applications médicales et études de recherche.

3. Concepts de base sur les séquences nucléiques et protéiques en bioinformatique

Les séquences nucléiques et protéiques représentées par des chaînes de caractères, avec un alphabet de 20 lettres pour les protéines (désignant les résidus d'acides aminés individuels) et un alphabet de 4 lettres pour les acides nucléiques (indiquant les bases individuelles dans les polymères d'ADN ou d'ARN). Les collections de séquences de bases de données internationales sont des ressources de premier ordre pour la recherche en biologie moléculaire. Nous présentons dans ce qui suit les définitions de plusieurs concepts aidant à la compréhension de la structure des séquences nucléiques et protéiques.

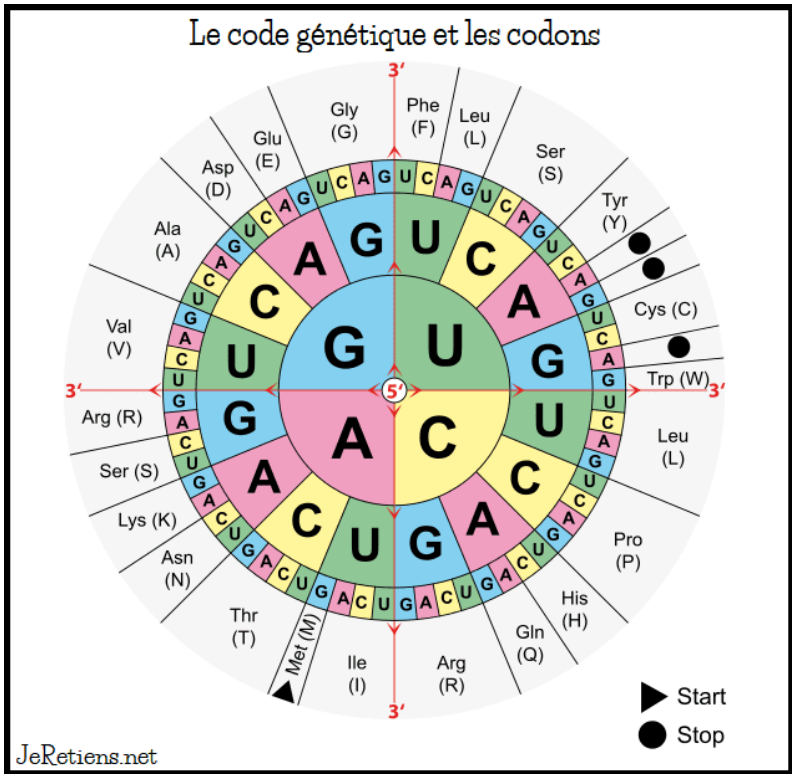

Figure 1: Présentation du code génétique et de ses codons de départ et d'arrêt sous une forme schématisée et originale [2].

Le gène

Les gènes sont les facteurs héréditaires des organismes eucaryotes, procaryotes et viraux. La génétique classique a démontré que les gènes sont des segments de chromosomes chez les eucaryotes . La génétique moléculaire a établi que les gènes sont des segments de molécule de l'ADN chez les bactéries et chez les virus à ADN [3].

Le gène est constitué d'un ensemble de nucléotides qui contient toutes les informations nécessaires pour transcrire un ARNm [4]. La figure 2 montre la structure d'un gène

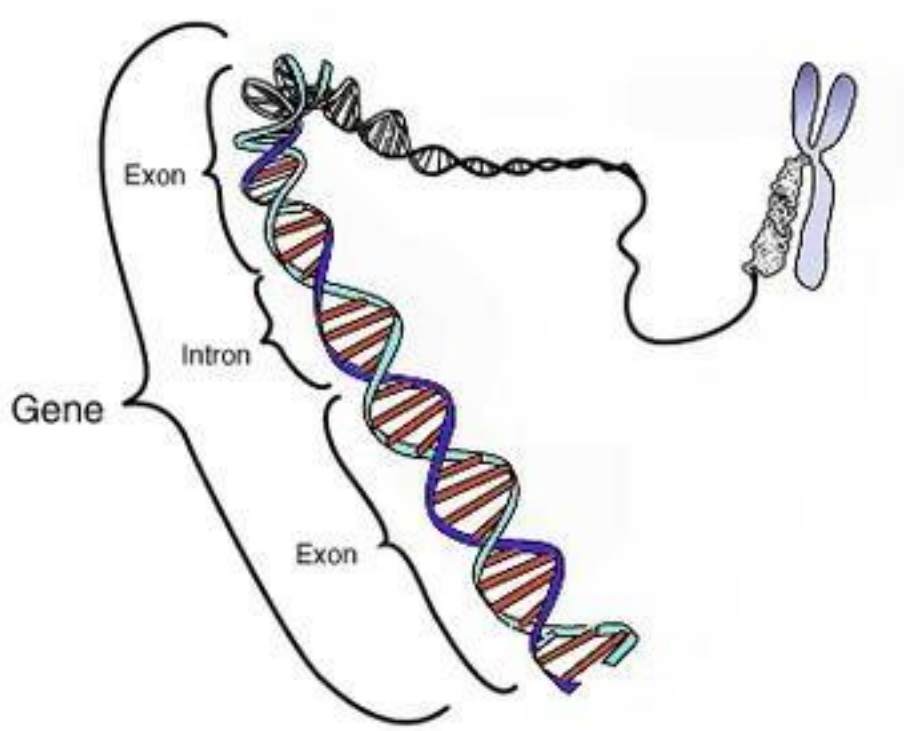

Figure 2 : La structure du gène [5].

Le Génome

Le génome d'une espèce représente la totalité de l'information génétique propre à celle–ci . la plus grande partie du génome d'une espèce eucaryote est nucléaire et portée par l'ADN chromosomique. Chez une espèce eucaryote, une petite partie du génome peut être extra-nucléaire (ADN mitochondrial ou chloroplastique) [6].

Le 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[7]. 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 [8]. La figure 3 représente la structure de chromosome.

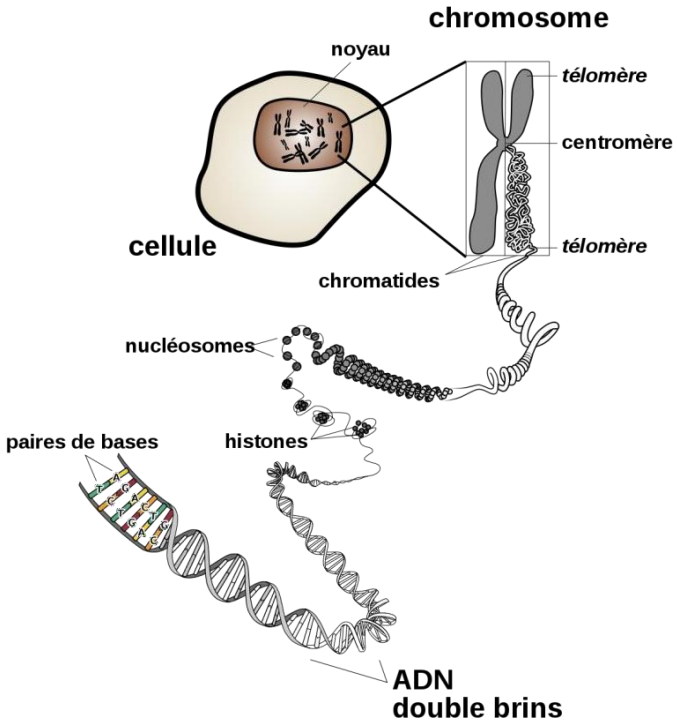

Figure 3 : La structure de chromosome [9].

Acide désoxyribonucléique

L' acide désoxyribonucléique est une molécule constituant le matériel génétique des êtres vivants. Chez les animaux et les plantes, il est un composant essentiel des chromosomes mais il est aussi rencontrée dans la mitochondries et les chloroplastes. Chimiquement, l'ADN consiste en deux chaines ou brins de poly-nucléotides dont les orientations sont opposées l'une à l'autre [10].

La double hélice d'ADN est repliée sur elle-même sous la forme de chromosomes et se situe dans le noyau de chaque cellule. La figure ci-dessous représente une double hélice d'ADN (Figure 4).

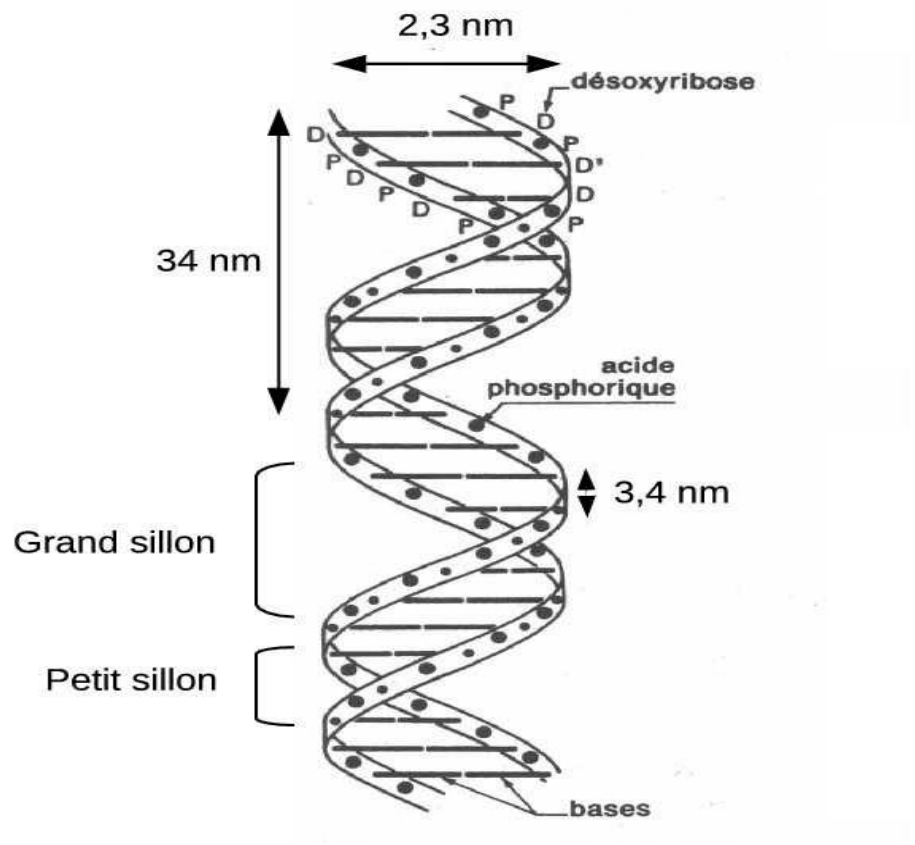

Figure 4: Représentation schématique d'une double hélice d'ADN [11].

Acide ribonucléique

L'ARN, ou acide ribonucléique, est construit à partir de l'ADN dans les cellules eucaryotes. Un segment d'ARN est très similaire à un segment d'ADN, les principales différences étant le sucre, qui est un ribose plutôt qu'un désoxyribose, et la thymine (T) qui est remplacée par l'uracile (U). La figure 5 représente la différence entre l'ADN et l'ARN , L'ARN est également simple brin dans la cellule et généralement de petite taille comparativement à l'ADN (50 à 5000 nucléotides) [12].

On a reconnu dans l'ARN cellulaire l'existence de trois fonctions qui jouent des rôles bien distincts : ARN messager, ARN ribosomial et ARN de transfert (ou ARN soluble) [13].

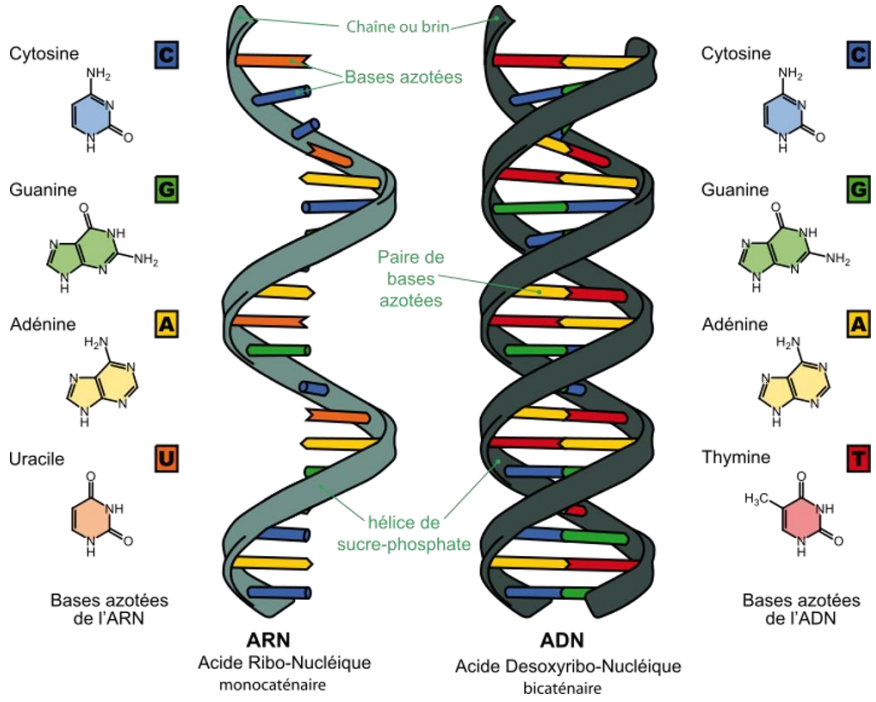

Figure 5 : La différence entre l'ADN et l'ARN [14].

Acides aminés

Unité de base ou monomère à partir duquel sont formées les protéines [15]. La formule générale d'une acide aminé (Figure 6) montre une fonction acide et une fonction acide (COOH) et une fonction amine (NH2) liées au même carbone. La chaine latérale ou radical R est propre à chaque acide aminé [16]. Il existe 20 acides aminés différents utilisés pour fabriquer les protéines [17].

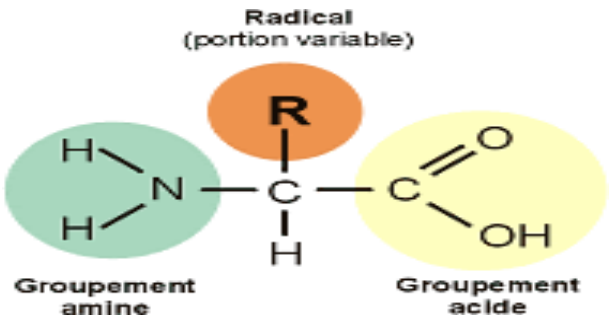

Figure 6 : La structure des acides aminés [18].

Protéines

Les protéines sont une classe de molécules biologiques. Ce sont des macromolécules de type polymère composées d'une ou plusieurs chaînes d'acides aminés (chaînes polypeptidiques).

En fonction de l'importance de la polymérisation et de la composition, nous pouvons distinguer différents types de protéines [19]. Voir figure (7)

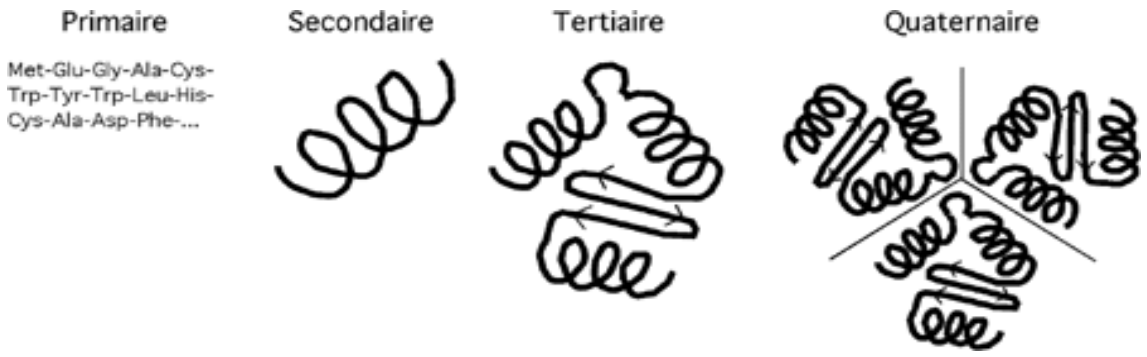

Figure 7: Les différentes structures des protéines [20].

4. Les Banques de données nucléiques et protéiques

Les banques de données sont un ensemble de données relatives à un domaine, organisées par traitement informatique, accessibles en ligne et à distance. Les données sont souvent stockées sous la forme d'un fichier texte formaté (respectant une disposition particulière). Nous pouvons distinguer deux types de banques de données de séquences ; des banques nucléiques [21], telles que **:**

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

Ainsi que des banques protéiques [22], telles que :

- **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 ProteinAnalysis System) et l'EBI. Celle-ci regroupe aussi des séquences annotées de la banque PIRNBRF ainsi que des séquences codantes, traduites de l'EMBL.

Les plus populaires sont :

- Ensembl (European Bioinformatics Institute / Wellcome Trust Sanger Institute).
- NCBI (National Cancer for Biology Information).
- UCSC (University of California Santa Cruz).

5. Fichier / Format FASTA

En bioinformatique et en biochimie, le format FASTA est un format textuel pour représenter des séquences nucléotidiques ou des séquences d'acides aminés (protéines), dans lesquels les nucléotides ou les acides aminés sont représentés à l'aide de codes à une seule lettre. Le format permet également aux noms de séquence et aux commentaires de précéder les séquences. Le premier caractère de la ligne de description est un symbole supérieur à (">"). Le format provient du progiciel FASTA [23], mais est maintenant devenu une norme quasi universelle dans le domaine de la bioinformatique.

La simplicité du format FASTA facilite la manipulation et l'analyse des séquences à l'aide d'outils de traitement de texte et de langages de script tels que le langage de programmation R, Python, Ruby et Perl. La figure 8 montre un exemple du format FASTA.

```
>gi|528476558|ref|NC_018928.2| Homo sapiens chromosome 17
TCCACTCACTGAGACAATAGACCCAAGCACATCAGCTCTGAGGCCTC
AGCACATCAGCTCTGAGGCCTCCACTCACTGAGACAATAGACCCAAG
TCACTGAGACAATAGACCCAAGCACATCAGCTCTGAGGCCTCCACTC
AACAGCTCTGAGGGCTCCACTCACTGAGACAACAGACCCAAGAACAA
>gi|528476670|ref|NC_018912.2| Homo sapiens chromosome 1
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
CCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAACCCTAA
ACCCTAACCCTAACCCTAACCCTAACCCAACCCTAACCCTAACCCTAACCCTAACCCTA
TAACCCTAACCCTAAACCCTAAACCCTAACCCTAACCCTAACCCTAACCCTAACCCCAA
```

Figure 8 : Exemple du format FASTA [24].

6. Méthodes de recherche de sous-séquences

Il existe une littérature abondante sur le sujet de la recherche de séquences d'ADN et de protéines, mais comme la quantité d'informations disponibles sur les séquences continue de se multiplier, le problème reste très pertinent aujourd'hui. La classe prééminente de méthodes pour aligner deux séquences est celle basée sur la programmation dynamique, une technique appliquée pour la première fois au problème par Needleman et Wunsch (1970) et sujette à de nombreux raffinements ultérieurs, notamment par Smith et Waterman (1981).

Le besoin d'une plus grande vitesse, en particulier lors de la recherche de correspondances dans une grande base de données de séquences, a conduit au développement d'autres outils logiciels tels que FASTA (Lipman et Pearson 1985 ; Pearson et Lipman 1988), BLAST (Altschul et al. 1990, 1997) et Cross_match (P. Green, inédit ; voir http://www.phrap.com). Dans chaque cas, une étape de recherche préliminaire basée sur les mots identifie les correspondances candidates probables, qui sont ensuite étendues à des alignements complets.

Ces dernières années, les algorithmes d'arbre de suffixes ont gagné en popularité en tant que méthodes de recherche de séquences ; un traitement approfondi du sujet est donné par Gusfield (1997). Delcher et ses collaborateurs (1999) décrivent un outil basé sur un arbre de suffixes qui est capable d'aligner les séquences de longueur du génome et également de localiser les SNP (Single Nucleotide Polymorphism), les gros inserts, les répétitions significatives, les répétitions en tandem et les inversions. Les méthodes d'arbre de suffixes sont intéressantes en raison de leur vitesse de recherche potentiellement rapide, mais elles peuvent dévorer une grande quantité de mémoire : Delcher et al. Pour une séquence de taille comparable au génome humain (environ trois giga bases), il faudrait une machine avec une quantité de mémoire exceptionnellement grande (au moins selon les normes actuelles) pour intégrer entièrement l'arbre de suffixes associé dans la RAM.

7. Conclusion

La tâche de rechercher des séquences a déjà conduit au développement d'un certain nombre de méthodes exhaustives de correspondance de chaînes, qui ont dépassé la capacité de nombreux postes de travail et ordinateurs centraux. La situation se détériore davantage à mesure que la taille des bases de données ou des fichiers objets de recherche augmente. Pour cela il existe d’autres types de méthodes qui ne sont pas exactes mais qui sont moins

gourmands et qui fournissent de bonnes solutions en un temps raisonnable. Ce type de méthodes sont appelées les métaheuristiques et nous les présenterons dans le chapitre suivant.

Chapitre 2 : Optimisation par métaheuristiques

1. Introduction

Les métaheuristiques sont des procédures de haut niveau conçues pour résoudre des problèmes d'optimisation dits difficiles. Ce sont en général des problèmes aux données incomplètes, incertaines, bruitées ou confrontés à une capacité de calcul limitée. Les métaheuristiques ont rencontré le succès dans une large variété de domaines [25]. Cela découle du fait qu'elles peuvent être appliquées à tout problème pouvant être exprimé sous la forme d'un problème d'optimisation de critère(s). Ces méthodes sont, pour la plupart, inspirées de la physique (recuit simulé), de la biologie (algorithmes évolutionnaires) ou de l'éthologie (essaims particulaires, colonies de fourmis). Dans ce qui suit, nous présentons brièvement les principales métaheuristiques d'optimisation. Nous les classons en deux catégories :métaheuristiques à solution unique et métaheuristiques à population de solutions.

2. Notions de base sur l'optimisation

Problème d'optimisation

Un problème d'optimisation se définit comme la recherche, parmi un ensemble de solutions possibles S (appelé aussi espace de décision ou espace de recherche), de la (ou des) solution(s) x^* qui rend(ent) minimale (ou maximale) une fonction mesurant la qualité de cette solution. Cette fonction est appelée fonction objectif ou fonction coût [26].

Lorsque nous voulons résoudre un problème d'optimisation, nous cherchons la meilleure solution possible à ce problème, c'est-à-dire l'optimum global. Cependant, il peut exister des solutions intermédiaires, qui sont également des optimums, mais uniquement pour un sous-espace restreint de l'espace de recherche : on parle alors d'optimums locaux. Cette notion est illustrée dans la figure 9. La seule hypothèse faite sur S est qu'il s'agit d'un espace topologique, i.e. sur lequel est définie une notion de voisinage.

Cette hypothèse est nécessaire pour définir la notion de solutions locales du problème d'optimisation. On peut alors définir un optimum local (relativement au voisinage V)

comme la solution x^* de S telle que $f(x^*) \leq f(x); \forall x \in V(x^*)$ [10].

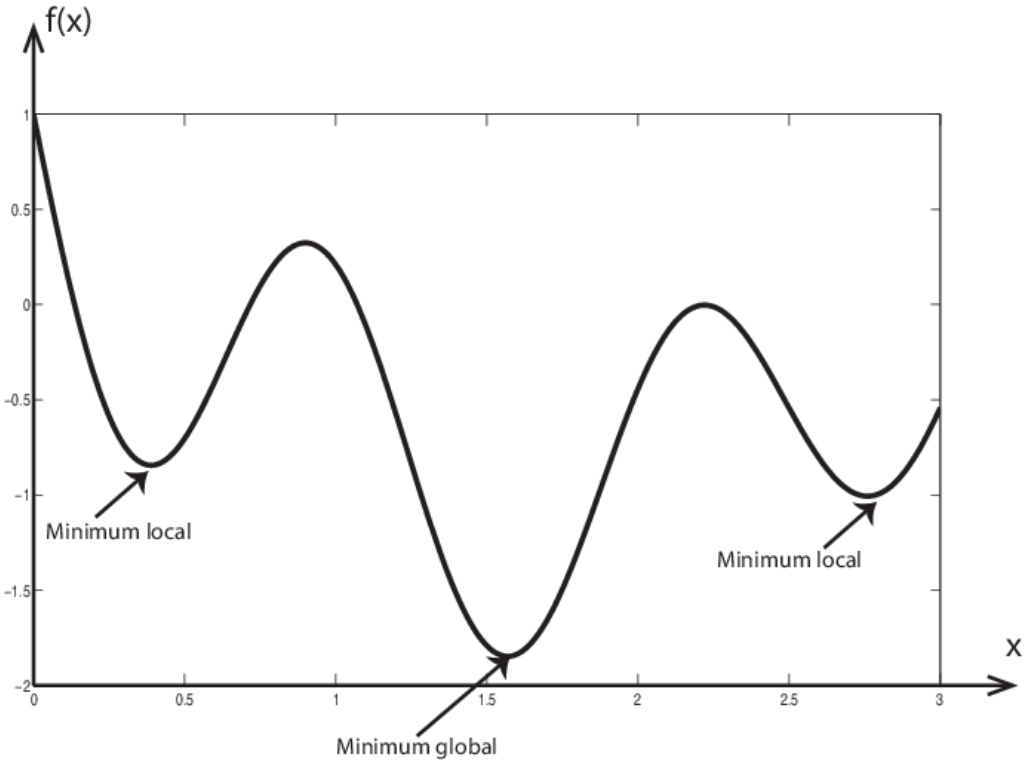

Figure 9 : La différence entre un optimum global et local [27].

3. Propriétés des métaheuristiques

Les métaheuristiques d'optimisation sont des algorithmes généraux d'optimisation applicables à une grande variété de problèmes. Elles sont apparues à partir des années 80, dans le but de résoudre au mieux des problèmes d'optimisation [28].

Les propriétés qui caractérisent la plupart des métaheuristiques sont :

- Les métaheuristiques sont des stratégies qui guident le processus de recherche.
- L'objectif est d'explorer efficacement l'espace de recherche afin de trouver des solutions quasi optimales.
- Ne nécessite pas le calcul de dérivées de fitness.
- Les techniques qui constituent les algorithmes métaheuristiques vont de simples procédures de recherche locale à des processus d'apprentissage complexes.
- Les métaheuristiques ne sont pas spécifiques à un problème donné.

- Ce sont des méthodes stochastiques, qui explorent l'espace selon une composante aléatoire.
- Ces méthodes sont inspirées de processus naturels.

- Les métaheuristiques utilisent des paramètres pour guider et optimiser la façon dont l'espace est exploré. La valeur idéale de ces paramètres est en général inconnue. Cela peut d'ailleurs dépendre du problème particulier
- Ces méthodes sont gourmandes en temps de calcul (mais on n'a pas d'autres choix). En revanche, elles se parallélisent assez facilement.
- Elles ne garantissent pas non plus la découverte de l'optimum global en un temps fini. Cependant, un grand nombre de problèmes réels n'est pas optimisable efficacement par des approches purement mathématiques, les métaheuristiques peuvent alors être utilisées avec profit.

En dernière analyse, il est parfois possible que le choix de la représentation des solutions, ou plus généralement des méthodes associées à la métaheuristiques, ait plus d'influence sur les performances que le type d'algorithme lui-même. En pratique, cependant, les métaheuristiques se montrent plus puissantes que les méthodes de parcours exhaustif ou de recherche purement aléatoire [29].

Concepts importants des métaheuristiques

Dans ce qui suit nous présentons les plus importants concepts à prendre en considération lors de la création d'une métaheuristique.

1. Une métaheuristique doit commencer par définir sa fonction objectif et une manière de représenter ses solutions c'est-à-dire comment la solution est codée dans l'algorithme et comment la qualité d'une solution est mesurée [26].

2. Une métaheuristique doit garder un équilibre entre les deux mécanismes principaux qui sont l'intensification et la diversification.

La diversification

Un processus d'exploration de l'espace de recherche, il permet de découvrir de nouvelles régions de l'espace de recherche ce qui permet d'éviter d'être bloqué sur des optima

locaux mais ce processus ne favorise pas la convergence de la métaheuristique [26]. Voir figure 10.

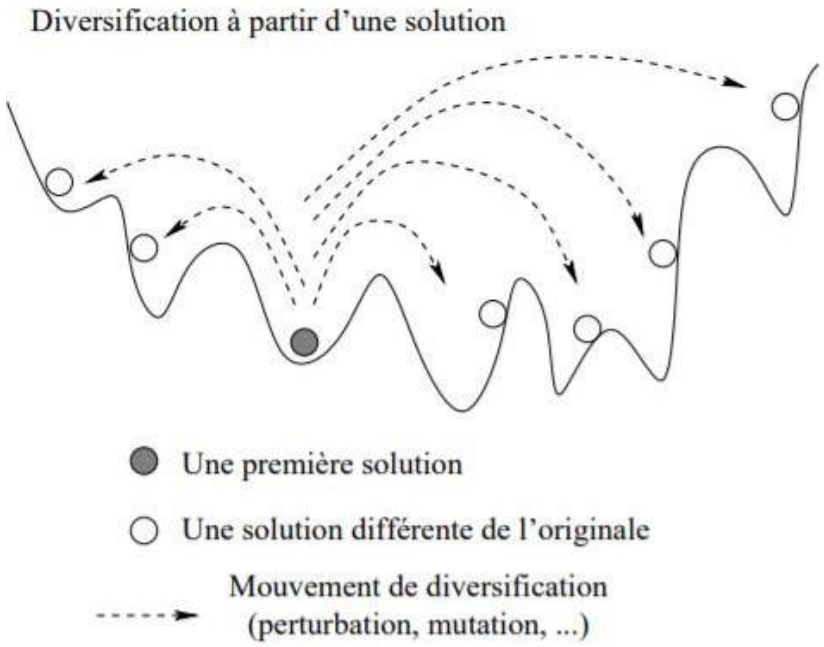

Figure 10 : Diversification

L'intensification

L'exploitation de l'information accumulée durant la recherche et concentration sur une zone précise où nous avons plus de chance de croiser l'optimum global [26]. Intensification signifie améliorer les meilleures solutions ou explorer leurs voisinages dans l'espoir de trouver des solutions meilleures, Voir figure (11)

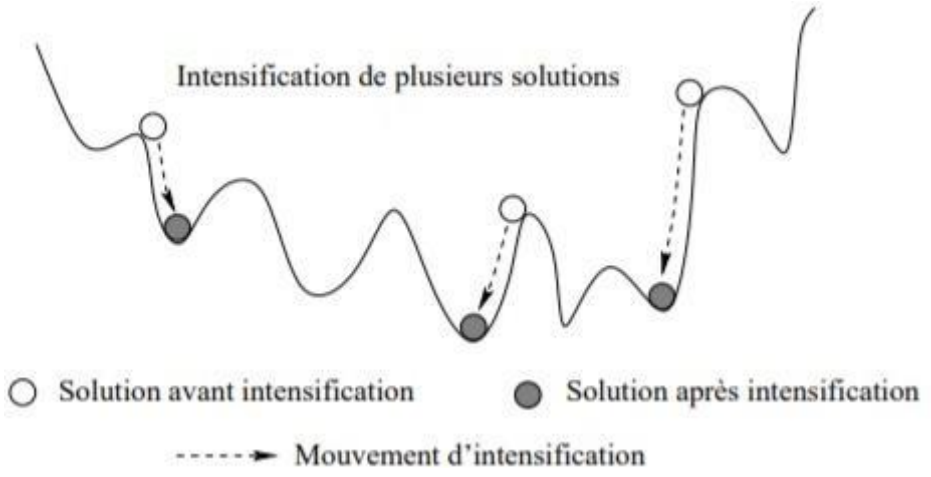

Figure 11 : Intensification

4. Domaines d'application des métaheuristiques

La métaheuristique est une discipline en plein essor qui entre en jeu dans beaucoup de domaines [30], comme dans la conception de circuits électroniques, la recherche opérationnelle, la biologie, mais aussi pour répondre aux besoins croissants des secteurs économique et industriel (maximisation des performances, minimisation des coûts) [31].

5. Classification des métaheuristiques

Il existe plusieurs façons de classer et de décrire les métaheuristiques. Selon les caractéristiques choisies, la figure 12 montre plusieurs classifications.

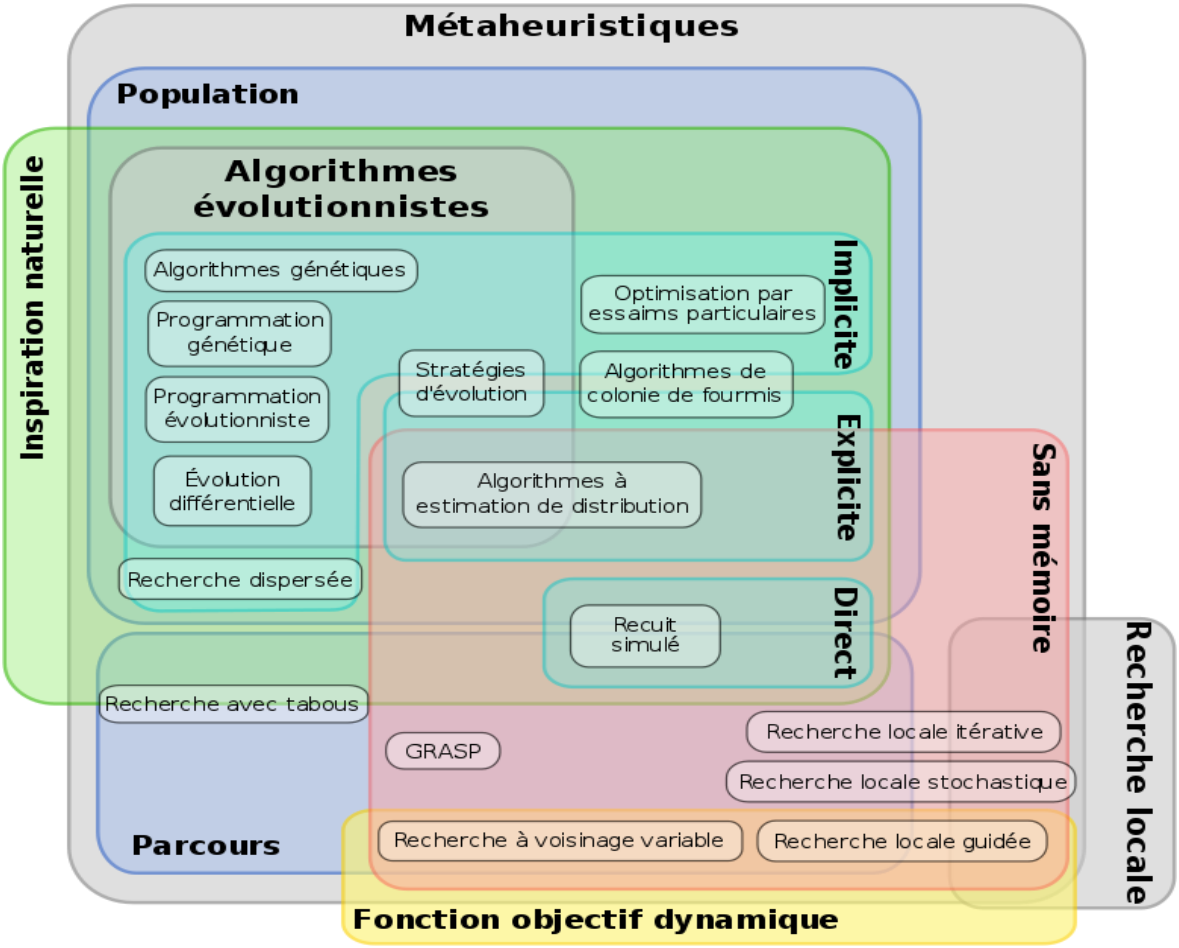

Figure 12 : Classification des métaheuristiques [32].

Recherche locale vs recherche globale

Il existe un grand nombre de métaheuristiques différentes, allant de la simple recherche locale à des algorithmes complexes de recherche globale. Ces méthodes utilisent cependant un haut niveau d'abstraction, leur permettant d'être adaptées à une large gamme de problèmes différents.

Les méthodes de recherche locale sont des algorithmes itératifs qui explorent l'espace X en se déplaçant pas à pas d'une solution à une autre. Une méthode de ce type débute à partir d'une solution $S_0 \in X$ choisie arbitrairement ou alors obtenue par le biais d'une méthode constructive.

Solution unique vs basée sur population

Les métaheuristiques à solution unique (les S-métaheuristiques), appelées aussi les méta- heuristiques de trajectoire et cette appellation est due à la trajectoire formée au cours du processus de recherche par l'évolution de la solution de démarrage. En effet, ces méthodes démarrent avec une solution unique et la remplace à chaque itération par une autre solution suivant un certain mécanisme, le processus de recherche s'arrête dès qu'un critère d'arrêt est atteint. Les méthodes de trajectoire les plus connues sont : la recherche tabou, le recuit simulé, la méthode de descente, la méthode GRASP, la recherche à voisinage variable.

Les métaheuristiques à base d'une population de solutions (les P- Métaheuristiques). Elles manipulent non pas une seule solution, mais un ensemble de solutions à chaque itération. Ce sont des méthodes qui font évoluer simultanément un ensemble d'individus (solutions) dans l'espace de recherche d'une itération à une autre d'une manière itérative. L'intérêt est d'utiliser la population comme facteur de diversité. Nous pouvons citer les algorithmes évolutionnaires inspirés de la génétique et de la théorie de l'évolution de C. Darwin et qui regroupe beaucoup d'algorithmes tels que l'algorithme génétique (GA) , les algorithmes d'intelligence en essaim tels que l'algorithme des colonies de fourmis (ACO) , les algorithmes à essaims de particules (PSO). Aussi nous avons les algorithmes basés sur la physique tels que l'algorithme de recherche basé sur la galaxie (GBSA), la recherche de système chargé (The charged system search) (CSS).

Méthodes Inspirées de la nature vs non inspirées de la nature

Une manière, plus intuitive, de classer les métaheuristiques consiste à séparer celles qui sont inspirées d'un phénomène naturel, de celles qui ne le sont pas. Les algorithmes génétiques et les algorithmes de colonies de fourmis sont inspirés de la nature tandis que l'algorithme de Recherche Tabou et l'algorithme de Recherche Locale Itérative sont non

inspirés de la nature, d'après les auteurs Blum et Roli, cette classification n'est pas significative :

- De nombreux algorithmes hybrides récents ne correspondent pas à ces deux classes (les deux classes en même temps).
- La difficulté de classer une méthode dans certains cas, par exemple, l'utilisation de la mémoire dans la recherche tabou n'est pas inspirée de la nature.

6. Les métaheuristiques les plus utilisées

Méthode de colonies de fourmis

La méthode de colonies de fourmis est apparue d'une constatation simple : les insectes sociaux, et en particulier les fourmis, résolvent naturellement des problèmes complexes. Un tel comportement est possible car les fourmis communiquent entre elles de manière indirecte par le dépôt de substances chimiques, appelées phéromones, sur le sol.

Le premier algorithme qui s'inspire de cette analogie a été proposé en 1996 par Colorni, Dorigo et Maniezzo [33]. Le but initial de cet algorithme était de résoudre le problème du voyageur de commerce. La figure 13 montre la structure de l'algorithme.

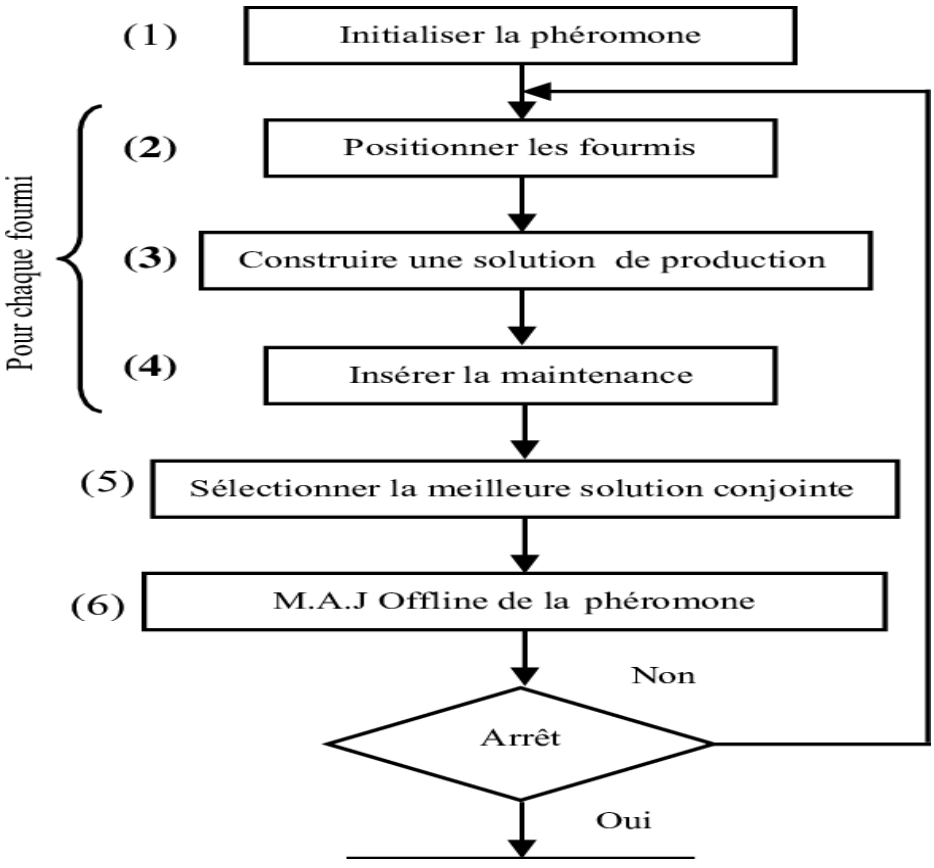

Figure 13 : Structure de l'algorithme de colonies de fourmis [34].

Méthode de recuit simulé

L'origine de la méthode du recuit simulé provient de la métallurgie, où, pour atteindre les états de basse énergie d'un solide, on chauffe celuici jusqu'à des températures très élevées, après on le laisse refroidir lentement. Ce processus est appelé le recuit. La méthode du recuit simulé a été développé simultanément par Kirk en 1983 [35] et Cerny en 1985 [36].

Cette méthode repose sur l'algorithme de Metropolis en 1953 [37]. Cet algorithme nous permet de sortir des minima locaux avec une probabilité élevée si la température T est élevée, et de conserver les états les plus probables pour des très basses températures. L'algorithme de Metropolis permet d'échantillonner la fonction objectif par le biais d'une distribution de Boltzmann de paramètre T. La figure 14 représente la structure de l'algorithme de recuit simulé.

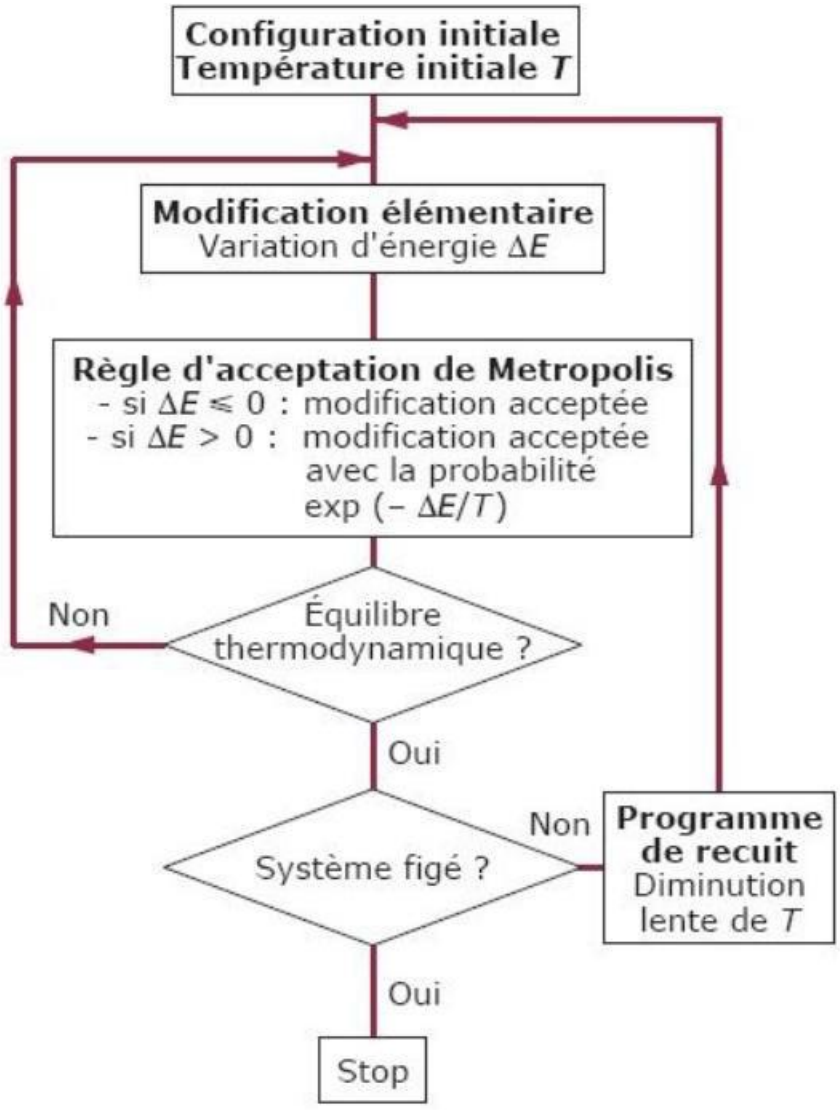

Figure 14 : La structure de l'algorithme de recuit simulé [38].

Recherche Tabou

La méthode de recherche Tabou est une métaheuristique originalement développée par Glover en 1986 [39] spécifiquement pour des problèmes d'optimisation combinatoire et qui permet de trouver d'une manière flexible un compromis entre la qualité de la solution et le temps de calcul. Le principe de cette méthode est d'ajouter au processus de recherche une mémoire flexible qui permet de mener une recherche plus "intelligente" dans l'espace des solutions.

Comme la méthode du recuit simulé, la méthode de recherche Tabou fonctionne avec une seule configuration courante, qui est actualisée au cours des itérations successives. La nouveauté ici est que, pour éviter le risque de retour à une configuration déjà visitée, on tient à jour une liste de mouvements interdits, appelée liste tabou. Cette liste contient m mouvements (t → s) qui sont les inverses des m derniers mouvements (s → t) effectués. La méthode modélise ainsi une forme primaire de mémoire à court terme. Dans sa forme de base, la méthode de recherche tabou présente l'avantage de comporter moins de paramètres que la méthode de recuit simulé. Cependant, la méthode n'étant pas toujours performante, il est souvent approprié de lui ajouter des processus d'intensification et/ou de diversification, qui introduisent de nouveaux paramètres de contrôle [40]. Voir figure 15.

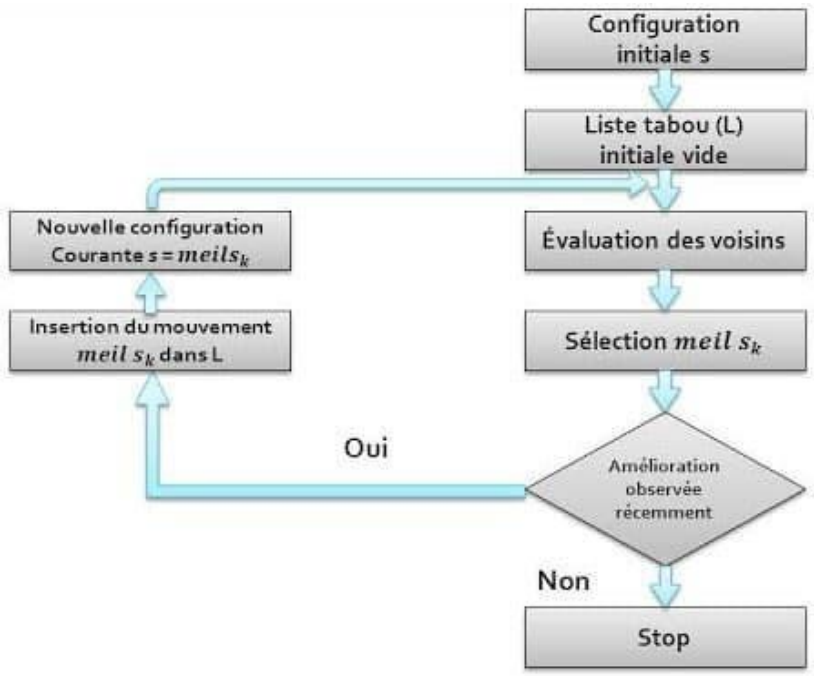

Figure 15 : Logigramme de la recherche de tabou [41].

Algorithmes génétiques

Les algorithmes génétiques (AG) sont des algorithmes de recherche inspirés des mécanismes de l'évolution naturelle des êtres vivants et de la génétique. Les premiers travaux ont été menés par Holland en 1975 [42].

Par analogie à la génétique, ces algorithmes cherchent à partir d'une population initiale les meilleurs individus. Pour constituer les parents appropriés donnant naissance à des meilleures descendances Enfants, parmi lesquelle seront tirée les solutions acceptables du problème traité. La figure 16 représente les étapes d'algorithme génétique.

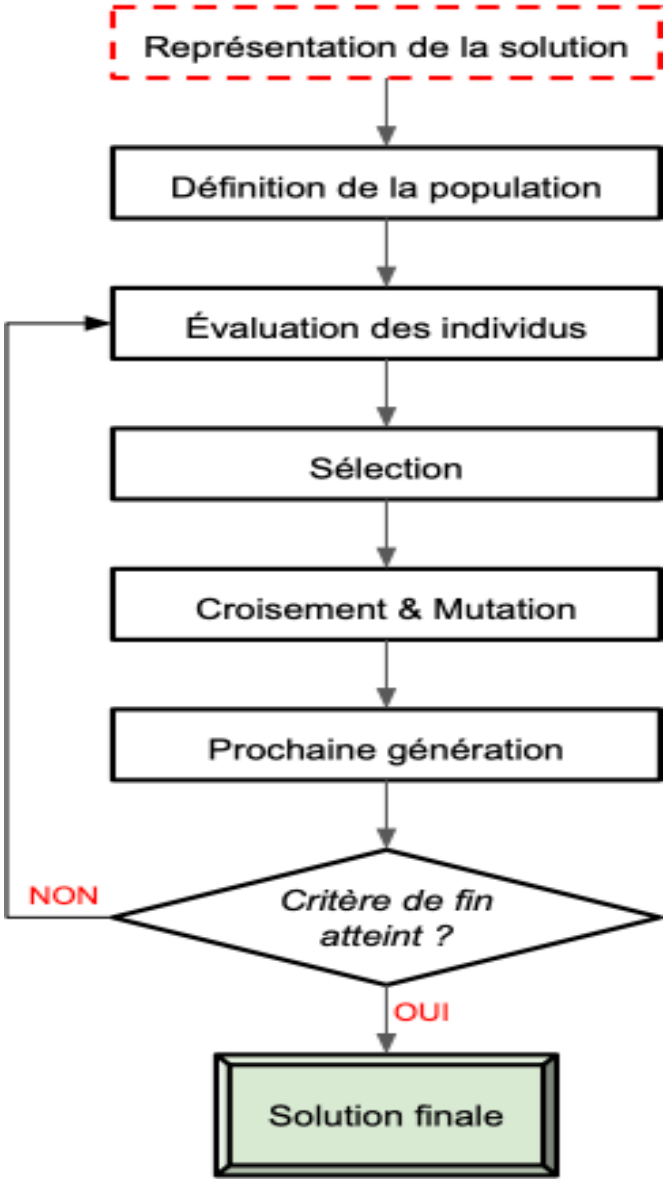

Figure 16 : Les étapes d'algorithme génétique [43] .

7. Quelques applications des métaheuristiques dans le domaine de la bioinformatique

Nous pouvons classer les problèmes de la bioinformatique dans le tableau 1, en donnant quelques approches métaheuristiques appliquées pour les résoudre.

- Alignement et comparaison des séquences génomiques et protéomiques
- Assemblage de fragments d'ADN
- Recherche et identification de gènes
- Profilage d'expression génique
- Prédiction de la structure
- Analyse phylogénétique

Tableau 1 : Quelques métaheuristiques représentatives appliquées aux principales tâches en bioinformatiques [44]

Tâche bioinformatique	**Métaheuristiques**
Comparaison et alignement de séquences	EA [45,46] MA [47] ACS [48] PSO [49]
Assemblage de fragments d'ADN	EA [50,51] SA [52] ACS [53]
Recherche et identification de gènes	EA [54,55]
Profilage d'expression génique	EA [56] MA [57,58] PSO [59]
Prédiction de structure	EA [60,61] MA [62] SA [63] EDA [64]
Arbres phylogénétiques	EA [65,66] SS [67] MA [68]

8. Conclusion

Les métaheuristiques sont des stratégies permettant de guider la recherche d'une solution optimale et elle s'appliquent à des espaces de recherche discrète ou continue, Les métaheuristiques sont souvent plus puissantes que des méthodes exactes dans les problèmes difficiles tell que le problème de recherche.

Dans le chapitre suivons nous allons implémenter un programme avec le langage python basé sur la méthode approcher la particule swarm optimisation (PSO) pour le but de trouvé la sous-séquence la plus longue et le comparé avec une autre méthode exacte la programmation dynamique (DP).

CHAPITRE 03 : Application de la métaheuristique PSO pour la recherche de la plus longue sous-séquences commune

1. Introduction

Dans ce chapitre, nous allons présenter notre contribution qui consiste à utiliser une métaheuristique pour résoudre le problème de la recherche de la plus longue sous-séquence commune d'une séquence donnée dans une autre séquence. Quelques métaheuristiques ont été déjà développées et présentées surtout celles basées sur les algorithmes génétiques. Dans notre travail nous avons choisi PSO, une métaheuristique qui a prouvé son efficacité dans plusieurs domaines et qui est considérée parmi les méthodes approchées les plus efficaces. Nous allons présenter un algorithme basé sur PSO appelé PSO-LCS adapté pour le traitement de la problématique sus-citée.

Ce chapitre est organisé comme suit : Dans une première partie nous allons présenter quelques définitions et notations. Puis nous allons décrire comment nous avons définit une solution donnée, l'espace de recherche, et comment nous avons formulé la fonction objectif. Ensuite, nous allons présenter l'implémentation algorithmique de l'approche proposée ainsi qu'une autre approche exacte. En suite nous allons expliquer le scenario d'évaluation expérimentale ainsi que la présentation et la discussion des résultats. Nous clôturons ce chapitre par une conclusion.

2. Définitions et notations

Nous considérons le problème de la recherche de la plus longue séquence commune d'une séquence de requête ***SeqRech*** de taille ***TailleSeqRech*** dans une autre séquence (stockée dans un fichier FASTA) sujet ***SEQUENCE*** de taille ***TailleSEQUENCE***.

Chaque sous-séquence dans **SEQUENCE** est étiquetée avec un entier ***i***, auquel nous le référons comme son indice.

Exemple :

Pour le problème de recherche d'une sous séquence ci-dessous, les indices sont indiqués au-dessous de chaque caractère dans la Figure 17:

Figure 17 : Les indices d'une séquence.

3. Formulation de la représentation d'une solution candidate

Une solution candidate ***SeqCand*** est représentée par une position (indice) dans la séquence sujet de recherche (SEQUENCE). Alors cette solution représente une sous séquence de **SEQUENCE** de taille **SeqRech.**

Exemple :

Pour le problème de recherche d'une sous séquence précédent (Figure 18) :

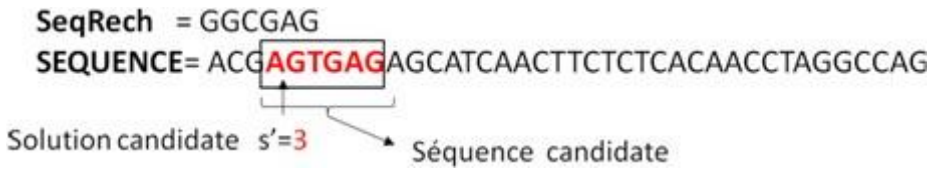

Figure 18 : La séquence candidate et solution candidate.

La solution candidate s'=3 concerne la sous séquence : AGTGAG

La solution candidate s'=5 concerne la sous séquence : TGAGAG

4. Espace de recherche

Selon la définition d'une solution candidate ***s'***, cette dernière doit être recherchée du début de la séquence sujet de recherche et sa taille moins la taille de la séquence recherchée.

$$s' \in [0 , TailleSEQUENCE - TailleSeqRech].$$

Exemple :

Pour le problème de recherche d'une sous séquence précédent :

L'indice de début de recherche est 0.

L'indice de fin de recherche est 39 – 6 = 33.

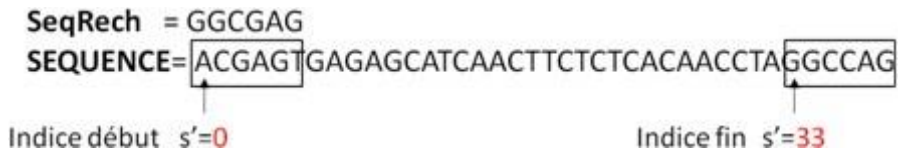

Figure 19 : L'indice de début et l'indice de fin de recherche.

5. Fonction objectif

La fonction objectif définit la qualité d'une solution donnée s'(solution candidate). Pour cela nous proposons pour notre problème de la recherche de la plus longue sous séquence commune une fonction objectif appelée ***EVALUER()*** représentée par le calcul d'un score entre la séquence recherchée ***SeqRech*** et une solution candidate ***SeqCand***de ***s'***.

Nous définissons le score entre ces deux séquences comme la taille de la plus longue sous séquence commune entre la séquence recherchée (***SeqRech***) et la solution candidate (***SeqCand(s')***).

Ce score est calculé en utilisant un algorithme exact (dans notre cas nous avons utilisé la fonction ***find_longest_match*** de Python).

EVALUER* (s')** revient à trouver la taille de la plus longue séquence commune entre ***SeqRech et***SeqCand(s')***.

Exemple :

EVALUER(7) revient à calculer le score précédemment définit entre **SeqRech**: GGCGAG et ***SeqCand(7)*** : GAGAGC.

Pour le problème de recherche d'une sous séquence précédent, (Figure 20).

Figure 20: Le score calculé à partir de seqCand et seqRech.

Le calcule donne une taille d'une sous-séquence commune égale à 3 entre les deux sous séquences (GGCC**AG** , G**AG**AGC).

→ ***EVALUER* (7) = 3**

6. PSO pour la recherche de la plus longue séquence commune

La solution proposée dans ce travail utilise l'algorithme PSO ayant une population (appelée essaim) de solutions candidates (appelées particules). Ces particules se déplacent dans l'espace de recherche selon quelques formules simples .

Les mouvements des particules (qui représentent de nouvelles solutions candidates) sont guidés par leur propre position la plus connue dans l'espace de recherche ainsi que par la position la plus connue de l'ensemble de l'essaim. Lorsque des positions améliorées seront découvertes, celles-ci viendront alors guider les mouvements de l'essaim.

Le processus est répété et, ce faisant, on espère, mais sans garantie, qu'une solution satisfaisante sera finalement découverte.Voir figure 21.

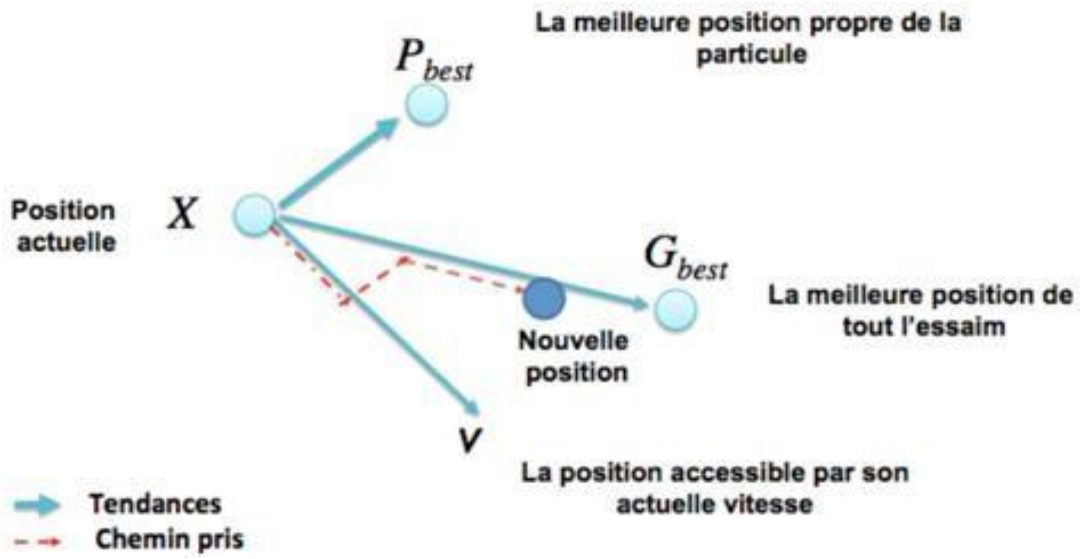

Figure 21 : Schéma de principe du déplacement d'une particule.

Au départ de l'algorithme, chaque particule est donc positionnée (aléatoirement ou non) dans l'espace de recherche du problème. Chaque itération fait bouger les particules en fonction de 3 composantes :

- Sa vitesse actuelle Vk,
- Sa meilleure solution Pi,
- La meilleure solution obtenue dans son voisinage Pg.

Cela donne l'équation de mouvement suivante :

Vk+1=omega Vk+b1(Pi-Xk)+b2(Pg-Xk).

Xk+1=Xk+Vk+1

Avec :

- Xk : est sa position actuelle
- omega : est son inertie
- b1 : est un nombre aléatoire tiré dans [0,φ1]
- b2 : est un nombre aléatoire tiré dans [0,φ2]

Malgré que le nombre de solutions possibles (solutions candidates) est limité dans notre problème, ce nombre est assez grand. Le parcours de toutes les solutions possibles n'est pas pratique pour chercher la meilleure solution. La technique de recherche adoptée par PSO comme d'autres métaheuristiques est d'essayer d'explorer quelques solutions candidates dans l'espace de recherche est de se rapprocher progressivement de la meilleure solution et l'atteindre (convergence) sans tester toutes les solutions (Voir Figure 22).

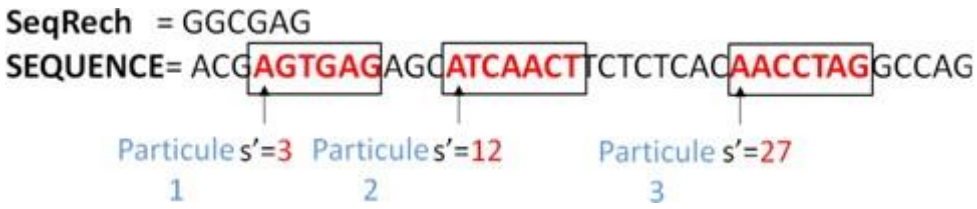

Figure 22 : Les solutions de séquences candidates.

7. Description des algorithmes

L'algorithme proposée PSO-LCS ainsi que l'algorithme PD-LCS sont présentées ci-dessous.

Algorithme PSO-LCS

Fonction EVALUER(solution):

Debut

|Retourner (*find_longest_match*(SeqRech, SeqCand(solution)))

Fin

Algorithme PSO-LCS

|c1<-1.0

|c2<-1.0

|w<-0.9

|MaxIter<-500

|TaillePop<-50

|TailleSeqRech<-190

```
|#//INITIALISATION //////////////////
|SEQENCE obtenir une séquence source
|TailleSEQENCETaille(SEQENCE)
|SeqRech<-récupérer une sous séquence aléatoire de taille TailleSeqRech depuisSEQENCE
|Gbest<-0
|GbestCost<-0
|Pour i de 0 à TaillePop-1 Faire
||      Vitesse[i] <-TailleSEQENCE/10
||      Position[i] <-generate_random_solution()
||      Lbest[i][0] <-Position[i]
||      Lbest[i][1] <-EVALUER(Position[i])
||      Si (Lbest[i][1]>GbestCost) Alors
||      |       Gbest <-Lbest[i][0]
||      |       GbestCost <-Lbest[i][1]
||      Fin Si
|FinPour|      #//////////////////////
|it<-1
|TantQue it<=MaxIter :
||      Pour i de 0 à TaillePop-1 Faire
||      |       Vitesse[i] <-w*Vitesse[i]+c1*random()*(Lbest[i][0]-
Position[i])+c2*random()*(Gbest-Position[i])
||      |       Position[i] <-Position[i]+Vitesse[i]
||      |       SI(Position[i]<0        or        Position[i]>TailleSEQENCE-
TailleSeqRech)Alors
||      |       |       Position[i] <-generer une solution aleatoire
||      |       FinSi
||      |       LbestTempCost<-EVALUER(Position[i])
||      |       Si (LbestTempCost >Lbest[i][1]) Alors
||      |       |       Lbest[i][0] <-Position[i]
||      |       |       Lbest[i][1] <-LbestTempCost
||      |       |       Si (Lbest[i][1] > GbestCost)
```

```
||      |       |       |       Gbest <-Position[i]
||      |       |       |       GbestCost <-Lbest[i][1]
||      |       |       |       Si (GbestCost<-TailleSeqRech)
||      |       |       |       |       break
||      |       |       |       FinSi
||      |       |       FinSi
||      |       FinSi
||      FinPour
||      Si (GbestCost<-TailleSeqRech)
||      |       break
||      FinSi
||      It<-it+1
|FinTantQue
|Ecrire('Meilleure Solution: ',Gbest , 'Score : ', GbestCost)
Fin
```

Algorithme PD-LCS

La programmation dynamique Inventée par le professeur Richard Bellman, elle permet de résoudre au moyen d'un ordinateur tout problème d'optimisation dont la fonction objectif se décrit comme la somme de fonctions monotones non-décroissantes des ressources. Concrètement, cela signifie que l'on va pouvoir déduire la solution optimale d'un problème à partir d'une solution optimale d'un sous problème

L'origine de l'algorithme PD-LCS

On sait depuis longtemps qu'en utilisant un arbre de suffixes, le LCSt d'un nombre arbitraire de chaînes peut être trouvé en temps proportionnel à la somme des longueurs des chaînes. Une structure de données importante dans la comparaison de chaînes est l'arbre des suffixes.

L'algorithme basé sur la programmation dynamique pour rechercher une sous séquence commune dans une autre séquence est décrit ci-dessous :

Algorithme: DP-LCS

Debut

```
|       TailleSeqRech<-190
|       SEQENCE<- obtenir une séquence source
|       TailleSEQENCE<-Taille(SEQENCE)
|       SeqRech<- récupérer une sous séquence aléatoire de taille TailleSeqRech depuis SEQENCE
|       # Variable pour stocker la taille de la plus longue séquence commune (LCS)
|       resultat <-0
|       # Variable pour stocker l'indice fin de LCS de SEQUENCE
|       dernier <- 0
|       # Matrice pour stocker le resultat de 2  lignes consecutives en meme temps
|       Longueur <- Tableau [2,TailleSEQENCE] initialisé par des 0
|       # Variable qui represente la ligne courante de la matrice
|       LigneCour <- 0
|       Pour i de 0 à TailleSEQENCE Faire
|       |       Pour j de 0 à TailleSeqRech Faire
|       |       |       Si (i <-0 ou j <- 0) Alors
|       |       |       |       Longueur[LigneCour][j]  <- 0
|       |       |       SinonSi (SEQENCE[i - 1] <- SeqRech[j - 1]) Alors
|       |       |       |       Longueur[LigneCour][j] <- Longueur[1 - LigneCour][j - 1]+1
|       |       |       |       Si (Longueur[LigneCour][j] > resultat) Alors
|       |       |       |       |       resultat <- Longueur[LigneCour][j]
|       |       |       |       |       dernier <-i – 1
|       |       |       |       |       Si (resultat<- TailleSeqRech) Alors
|       |       |       |       |       |       Break
|       |       |       |       |       FinSi
|       |       |       |       FinSi
|       |       |       Sinon
|       |       |       |       Longueur[LigneCour][j] <- 0
|       |       |       FinSi
|       |       FinPour
```

```
|       |       Si (resultat<- TailleSeqRech) Alors
|       |       |       Break
|       |       FinSi
|       |       LigneCour <-1 – LigneCour
|       FinPour
|       Si (resultat <- 0) Alors
|       |       Retourner (-1)
|       FinSi
|       Ecrire('Meilleure Solution: ',dernier - resultat + 1 , 'Score : ', resultat)
Fin
```

8. Scenario d'évaluation expérimentale et performances mesurées

D'abord L'algorithme proposé a été implémenté en langage Python. Afin de comparer les performances de notre algorithme proposé, un algorithme exact basé sur la programmation dynamique qui est également implémenté pour résoudre le problème de la plus longue sous-séquence commune. L'algorithme exact est présenté dans la section [7.2.1.]. En reprenant les notations suivantes : l'algorithme proposé est référencé comme PSO-LCS et l'algorithme servant de repère est référencé DP-LCS.

PSO-LCS proposé est testé par rapport à DP-LCS sur des instances choisies aléatoirement de tailles différentes de la séquence **SEQUENCE**. La séquence source de recherche considérée **(SÉQUENCE)** est la première partie du chromosome humain 1 (LT = 250 M bases) (Genome Reference Consortium, UCSC). La sous-séquence recherchée (***SeqRech)*** est un segment choisi au hasard de 190 bases de ***SEQUENCE***. Nous avons considéré 5 modalités de la taille de la séquence sujette de recherche (***SEQUENCE***) : 200.000, 400.000, 600.000, 800.000, 1.000.000, 1.200.000, 1.400.000 et 1.600.000 bases. Toutes les expériences ont été menées sur un ordinateur de 8 Go de RAM et un processeur i3…… . Vu que l'algorithme proposé PSO-LCS n'est pas déterministe, 30 exécutions sont effectuées pour la même expérience et le résultat d'une expérience est considéré comme le moyenne de ces 35 exécutions.

A chaque expérience, nous mesurons le temps nécessaire pour trouver la plus longue sous-séquence commune et sa longueur. Pour **PSO-LCS**, nous mesurons aussi le

nombre d'itérations nécessaire pour atteindre la meilleure solution (pour trouver toute la sous-séquence ***SeqRech***. L'erreur standard est aussi considérée lors de la présentation des résultats de chaque expérience.

9. Paramètres expérimentaux

Les valeurs des paramètres expérimentaux utilisées dans les différentes expériences d'évaluation des algorithmes PSO-LCS et PD-LCS sont indiquées dans le Tableau 2.

Tableau 2 : Valeurs des paramètres expérimentaux.

Paramètre (unité)	**Valeur**
Longueur de la séquence recherchée (Bases)	190
Taille de la séquence source de recherche (Bases)	200.000 à 1.600.000
Nombre de répétitions	35
Taille de la population (Particules) PSO	50
Nombre d'itérations maximal dans PSO	500
Coefficient c1 de PSO	1.0
Coefficient c2 de PSO	1.0
Inertie w de PSO	0.9

10. Résultats et discussion

Après l'exécution de notre code de PSO-LCS nous avons obtenus ces résultats présentées dans le tableau 3.

Tableau 3 : Les résultats de temps après l'exécution du code de PSO-LCS.

Rép	200000	400000	600000	800000	1000000	1200000	1400000	1600000
1	41,888994	16,099787	13,781117	34,703111	22,661276	64,876921	25,498218	33,517457
2	34,717084	65,971176	0,330806	84,764351	0,082941	0,18691	37,179988	27,827402
3	9,592519	0,230868	0,412738	22,6113	16,990252	69,883128	25,205832	37,634664
4	16,048795	0,278839	55,561121	0,270869	34,335116	14,672573	25,026833	21,010962
5	59,824678	7,224851	0,344792	0,269819	56,241258	32,093821	42,71329	72,421124
6	20,107469	21,550618	25,570329	85,571926	0,252854	0,18991	27,729249	0,049963
7	19,766681	0,282857	14,919417	0,609614	33,361978	24,648108	17,636053	62,04198
8	12,961576	18,637299	0,198907	35,017926	59,691558	20,686503	16,455116	36,851754
9	26,746625	0,220871	57,431046	44,670398	47,19447	44,603963	58,158761	33,746912
10	18,150718	22,998806	0,302824	0,333781	54,392561	31,308725	45,448511	36,422581
11	19,398853	0,221874	29,254231	35,685552	8,350207	63,446706	58,527096	37,349598
12	17,438019	19,520824	21,670566	79,691245	0,163928	0,242879	41,092051	38,495914
13	20,546217	0,169897	0,245878	28,530612	0,102961	25,402584	0,097943	66,067817
14	18,58534	0,28983	20,028508	0,564694	0,223863	28,313001	32,810375	70,673421
15	17,739806	18,652318	41,063446	42,273728	0,182876	36,379265	64,679185	29,789073
16	22,169283	52,178051	0,229852	0,146916	15,708264	27,702219	60,371428	63,99831
17	63,11182	20,903032	18,925123	34,785048	25,658659	71,26736	46,798446	0,05794
18	21,953408	63,743456	24,094197	32,281454	66,081334	22,76993	27,224404	61,069893
19	11,794235	50,027299	25,564335	15,021381	0,262848	38,520286	24,968741	38,715926
20	7,840523	0,39777	28,013935	38,950633	70,993389	68,026393	0,180919	60,524758
21	6,014571	5,283969	25,876157	56,139774	30,848318	26,22511	28,354066	64,18783
22	21,656596	24,638847	15,700992	52,90765	54,797088	29,705111	24,659141	41,039215
23	20,241407	23,638461	22,328184	59,325985	16,7679	20,605292	58,991376	0,165904
24	19,737669	18,022682	0,210851	0,218895	35,098147	18,090148	11,954159	30,098889
25	20,995959	47,772617	51,622377	29,200221	54,203329	33,429467	27,725247	33,406983
26	14,397742	23,335612	48,788988	0,318809	36,697361	57,75654	31,946257	24,618848
27	0,038982	65,014705	35,074881	16,324581	0,168922	42,354718	33,996518	31,636284
28	6,646189	29,002374	19,932561	63,055828	66,526238	0,135919	66,573522	52,174787
29	16,516546	16,023806	0,28983	60,781114	70,626163	38,753037	42,819364	41,110395
30	25059597	7,496695	0,09992	20,036503	0,139919	65,747485	51,834596	38,512175
31	14,77452	23,227698	41,713091	56,187764	0,201883	25,619302	58,057553	58,617848
32	0,23388	8,764969	0,384801	18,167578	23,687645	58,070878	57,978875	33,369284
33	19,863603	13,424296	18,92614	56,682507	31,301389	22,218349	32,277844	58,837624
34	63,235743	0,370797	0,181887	19,425855	22,169691	27,86433	19,435988	21,317237
35	18,163579	15,010394	18,417419	31,312005	8,885022	74,992886	16,239221	66,072401

Après l'exécution de notre code de DP-LCS nous avons obtenus ces résultats présentées dans le tableau 4.

Tableau 4 : Représente les résultats de temps après l'exécution du code de DP-LCS.

Rép	200000	400000	600000	800000	1000000	1200000	1400000	1600000
1	7,274532	17,137585	26,758631	148,873604	4,719332	169,402748	58,042518	207,331632
2	3,847441	3,896746	0,116959	158,619557	0,168905	0,19389	62,026534	172,477527
3	15,581043	0,084952	0,105943	96,295788	68,784811	18,16084	25,048589	25,672402
4	21,211833	0,07598	9,57853	0,134921	132,373781	89,167481	55,663446	119,210854
5	3,965728	51,573417	0,100923	0,28986	113,778688	58,741889	13,678919	240,530907
6	21,264919	12,18799	24,545941	86,337479	0,223877	0,19489	126,382656	0,258938
7	25,169543	0,080934	37,408519	0,228868	119,818525	94,927883	130,275732	28,945521
8	15,769032	13,412326	0,10792	31,387982	15,54612	64,803199	56,497966	159,917091
9	22,956852	0,071959	16,035802	27,773838	57,867887	14,657593	91,829661	126,533884
10	21,729514	11,219565	0,10694	0,162904	132,683539	124,368911	184,25023	153,402249
11	9,224732	0,071961	24,418994	145,347603	54,711916	15,006393	6,009555	169,070143
12	20,313326	41,678093	25,234502	39,500736	0,164905	0,186871	29,129228	64,979466
13	21,131857	0,072959	0,109915	130,404226	0,157891	26,087059	0,397773	52,179657
14	9,29167	0,083974	60,466453	0,162885	0,153914	143,740009	171,185007	56,916416
15	23,142748	9,693418	55,904955	720,884281	0,15791	78,461472	105,641029	101,968576
16	8,714979	53,064582	0,099944	0,131923	59,989967	23,781456	188,275588	214,334007
17	2,596512	28,587601	13,573193	71,689876	59,299582	153,387952	18,45308	0,236845
18	5,650781	2,558534	17,867751	71,530991	13,355363	56,71981	62,612718	175,397034
19	25,434412	52,606845	56,084808	50,553024	0,155934	56,409421	140,172434	23,652521
20	21,573625	0,081978	14,349924	65,815249	99,732822	145,399859	0,227975	222,917637
21	8,567086	18,935159	36,871871	4,078664	62,602232	136,522395	107,362401	209,371913
22	25,470371	23,062976	18,889143	2,713424	20,22764	66,611127	65,881991	114,199348
23	19,733658	17,956701	69,516122	17,536942	52,972829	55,462904	197,204218	0,257855
24	14,747563	12,708711	0,113958	0,126928	24,176486	106,092938	86,148784	21,526652
25	11,829216	9,161722	1,660047	73,217003	138,331752	87,939519	65,391297	85,378902
26	11,152605	17,448014	12,441884	0,125906	117,884643	160,841963	112,576923	133,791097
27	0,046973	5,828658	5,274952	1,459162	0,153913	165,3532	134,494011	183,244555
28	10,499975	15,124326	15,612046	5,37192	6,130483	0,184895	19,764058	214,240347
29	4,142624	29,129315	0,09992	18,347477	58,384788	40,495008	167,275589	155,114473
30	17,174249	6,025565	63,082816	52,815039	0,160931	125,847926	171,950883	64,154895
31	7,861514	13,718109	68,809507	111,969752	0,156912	17,270627	196,131415	90,644119
32	11,351489	2,242692	0,105941	82,017953	22,170283	170,599572	155,326682	173,935748
33	16,920317	10,153154	60,622204	110,872402	111,291228	67,597705	107,991399	203,562712
34	25,128586	0,074957	0,122933	21,905434	70,353897	128,140458	59,253132	186,199026
35	10,464997	22,341185	15,744969	23,031813	54,559206	6,325372	136,222539	211,021518

Nous avons représenté les résultats précédents dans le graphique suivant, voir figure 23, dans le but de comparer les deux méthodes mentionnées ci-dessus.

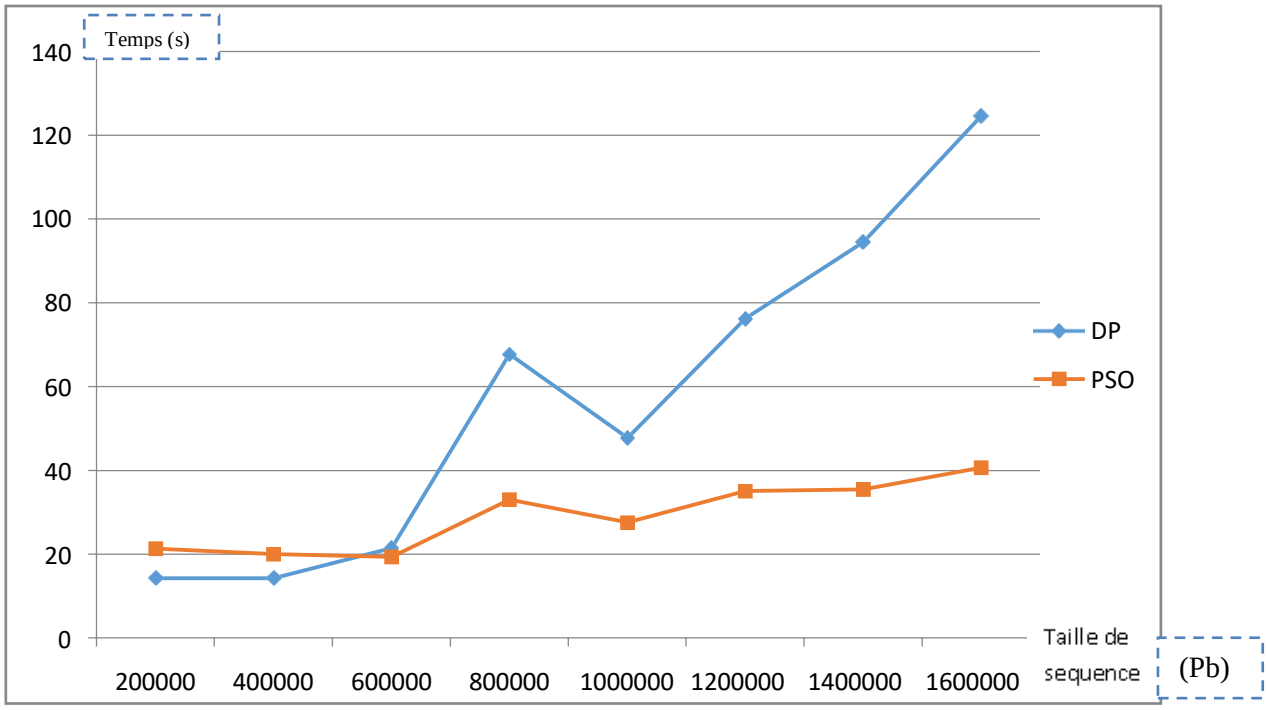

Figure 23 : Graphique représente une comparaison d'évolution de temps de PSO et DP en fonction de taille de séquence**.**

La figure 23 montre une comparaison des deux algorithmes DP-LCS et PSO-LCS proposé. Les deux graphiques représentent le temps nécessaire (en seconde) pour trouver la sous séquence recherchée en fonction de ta taille de la séquence source. Comme nous pouvons le constater, l'algorithme exact DP-LCS montre de meilleurs résultats pour des séquences de petites tailles allant de 100000 bases à 600000 bases. Ces observations sont expliquées par le fait que pour les séquences de petites tailles, la recherche ne nécessite pas beaucoup de temps pour trouver la sous séquence commune. Cependant le temps de recherche augmente exponentiellement en augmentant la taille de la séquence source de recherche, chose qui est illustrée par le graphique qui correspond à DP-LCS. Contrairement à DP-LCS, malgré que PSO-LCS montre des temps plus faibles au début de

la recherche, il nécessite moins de temps pour des séquences de taille plus grande (à partir de 600000 bases) et l'évolution de la courbe du graphique PSO-LCS suit une évolution linéaire. C'est grâce au comportement adopté par PSO et la formulation proposée de la fonction objectif (qui aide à définir la qualité d'une solution donnée) que la recherche converge rapidement vers la solution optimale.

11. Conclusion

Dans ce chapitre nous avons décrit un algorithme approché efficace qui utilise la métaheuristique PSO pour traiter le problème de la recherche de la plus longue sous-séquence commune dans une séquence source. Cet algorithme effectue des recherches rapides sur de grandes séquences d'ADN. Nous avons comparé ces performances avec un algorithme de recherche exact basé sur la programmation dynamique sur plusieurs scénarios d'évaluation. Les résultats ont montré l'efficacité de l'algorithme proposé est la puissance des métaheuristiques pour des problèmes de la recherche de la plus grande sous-séquence commune dans des séquences de grande taille.

Conclusion et Perspective

Conclusion

L'étude présentée dans ce modeste travail est consacrée à l'apport de l'utilisation des métaheuristiques dans la recherche de la plus longue sous-séquence commune d'une séquence.

L'objectif de ce travail c'est de montrer l'utilité et la façon dont les méthodes métaheuristiques peuvent être utilisées pour résoudre le problème de la recherche de la sous séquence commune. Nous nous sommes tout d'abord intéressés à présenter les notions de base sur les séquences nucléiques et protéiques ainsi que les méthodes de recherche de la plus longue séquence commune les plus utilisées. Nous avons ensuite présenté brièvement les principales métaheuristiques d'optimisation, leurs caractéristiques, leurs classifications et le principe de leur fonctionnement. Vu les limites des méthodes algorithmiques exacts dans le traitement des grandes masses de données telles que les séquences nucléiques, L'implémentation de l'application est présentée dans le troisième chapitre durant lequel nous avons deux algorithmes le PSO est le DP en se basant sur des résultats après plusieurs simulations, ces dernières confirment l'efficacité de notre algorithme avec un gain moyen en termes de temps d'exécution. Notre contribution consiste à proposer un algorithme basé PSO pour rechercher la plus longue séquence commune. Nous avons comparé les performances de l'algorithme proposé en termes de temps avec un autre algorithme exact basé sur la programmation dynamique. Les résultats ont montré une efficacité supérieure de l'algorithme proposé pour les longues séquences. Nous estimons que l'utilisation des métaheuristiques pour la résolution des problèmes bioinformatiques n'est pas assez explorée ce qui constitue en soi un axe de recherche important.

Références Bibliographiques

Références Bibliographiques

[1] Ichipro , (https://ichi.pro/fr/bioinformatique-et-biologie-computationnelle-quoi-pourquoi-comment-25762738737291) (consulté le 10 mai 2021).

[2] (https://jeretiens.net/comment-les-cellules-lisent-elles-les-genes/) (consulté le 10 mai 2021).

[3] Paulette Van Gansen , Henri Alexandre.(2004).BIOLOGIE GENERALE 4eme édition , Dounod,paris,ISBN 2 10 049026 5.

[4] Dr. Abdelhakim Aouf.(2015).Cours de Biologie Moléculaire et Génie Génétique

[5] (https://fr.sawakinome.com/articles/biology/unassigned-5835.html) (consulté le 10 mai 2021).

[6] Jean –Louis Serre , Louise Blottiére.(2013).MAXI FICHES ,Génétique, Dunod, paris, ISBN 978-2-10-058485-7.

[7] SERIBLI Houssam Eddine.(2015).Développement et Implémentation d'un Solveur Bio- inspiré pour l'Alignement de Séquences Biologiques.

[8] SERIBLI Houssam Eddine.(2015).Développement et Implémentation d'un Solveur Bio-inspiré pour l'Alignement de Séquences Biologiques.

[9] (https://bio.m2osw.com/gcartable/chromosomes.htm) (consulté le 14 mai 2021)

[10] Paulette Van Gansen , Henri Alexandre.(2004).BIOLOGIE GENERALE, Dunod, Paris , ISBN 2 10 049026 5.

[11] (https://www.researchgate.net/figure/FIG-B6-A-gauche-representation-schematique-dune-double-helice-dADN-A-droite_fig31_30513465) (consulté le 14 mai2021).

[12] QUINKAL, Isabelle.(2003). Quelques termes-clef de biologie moléculaire et leur définition.INRIA Rhône-Alpes.

[13] M. LAMOTTE , PH . L'HERITIER .(1968).BIOLOGIE GENERALE 2 Lois et Mécanismes De l'hérédité , Doin-Deren et Cie , 8, Place de l'Odéon, Paris.

[14] (https://theconversation.com/comment-fonctionnent-les-vaccins-a-arn-et-a-adn-125267) (consulté le 14 mai 2021).

[15] Bill Indge .(2004). LA BIOLOGIE DE A à Z ,1 100 entrées , des exemples et des conseils pour réviser , Dunod, Paris , ISBN 2 10 007411 3.

[16] B.AUGéRE , T , Darribére, J.M.Dupin, C.Escuyer, J.F.Fogelgesang, C. Van Der Rest. (2013). BIOLOGIE tout-en-un.Dunod, Paris , ISBN 978-2-10-0599922-6.

[17] QUINKAL, Isabelle.(2003). Quelques termes-clef de biologie moléculaire et leur définition. INRIA Rhône-Alpes.

[18] (https://bio.m2osw.com/gcartable/biomoleculaire/acides_amines.htm) (consulté le 17 mai 2021).

[19] Coralie Bouget. (2020).Protéines et acides aminés : utilisations par les sportifs et conseils à l'officine. Sciences pharmaceutiques.

[20] (https://biochimiedesproteines.espaceweb.usherbrooke.ca/2.html) (consulté le 18 mai 2021).

[21] C. BEROUD.(2011). Bases de données et outils bioinformatiques utiles en génétique. URL http://campus.cerimes.fr/genetique medicale/enseignement /genetique28 /site/html/cours.pdf.Collège National des Enseignants et Praticiens de Génétique Médicale, Université Médicale Virtuelle Francophone.

[22] C. BEROUD.(2011). Bases de données et outils bioinformatiques utiles en génétique, Collège National des Enseignants et Praticiens de Génétique Médicale,Université Médicale Virtuelle Francophone.

[23] William R. Pearson .(2011). The FASTA program package, 29 p.

[24] (https://www.researchgate.net/figure/Example-of-a-hypothetical-FASTA-file-content_fig1_315673633) (consulté le 18 mai 2021).

[25] Ahmed Nasreddine Benaichouche.(2014). Conception de métaheuristiques d'optimisation pour la segmentation d'images. Application aux images IRM du cerveau et aux images de tomographie par émission de positons .

[26] AHMIA Ibtissam.(2019).Une nouvelle métaheuristique pour les probléme d'optimisation combinatoire : La Monarchie Métaheuristique, Université Des Sciences Et De La Technologie.

[27] (https://www.researchgate.net/figure/Difference-entre-un-optimum-global-et-des-optima-locaux_fig1_278644845) (consulté le 20 mai 2021).

[28] Z. Huang, X. Miao, and P. Wang. (2007).A revised cut-peak function method for box constrained continuous global optimization. Applied Mathematics and Computation, 194 :224–233.

|29] Chopard B.(2006). Méthodes et Heuristiques d'Apprentissage et d'optimisation. Cours Université de Genève. septembre.

[30] Ilhem Boussaid.(2013). Perfectionnement de métaheuristiques pour l'optimisation continue. Autre. Université Paris-Est, Français. ffNNT : 2013PEST1075ff. fftel-00952774f.

[31] I.Boussaïd, A. Chatterjee, P. Siarry, & M. Ahmed-Nacer.(2013). A comparative study of modified bbo variants and other metaheuristic optimization techniques for the optimal power allocation in wireless sensor networks. In Amitava Chatterjee, Hadi Nobahari, & Patrick Siarry, editors, Advances in Heuristic Signal Processing and Applications, pp. 79–110. Springer Berlin Heidelberg,ISBN 978-3-642-37879-9.

[32] (https://www.researchgate.net/figure/Classification-des-metaheuristiques-Source-Wikipedia-auteur-Johann-Dreo_fig22_47694117) (consulté le 27 mai 2021).

[33] M. Dorigo, V. Maniezzo, and A. Colorni.(1996). Ant system : optimization by a colony of cooperating agents. IEEE Trans. on Man. Cyber. Part B, 26 :29–41.

[34] (https://www.researchgate.net/figure/Ordonnancement-conjoint-Production-Maintenance-par-la-strategie-integree-341_fig2_29621274) (consulté le 01 juin 2021).

[35] S. Kirkpatrick, C. Gelatt, and M. P. Vecchi. (1983).Optimization by simulated annealing. Science, 220 :671–680.

[36] V. Cerny.(1985).Thermodynamical approach to the traveling saleman problem : an efficient simulation algorithm. J. of Optimization th. and applications, 45 :41–51.

[37] N. Metropolis, M. Rosenbluth, A. Teller, and E. Teller. (1953).Optimization by simulated annealing, equation of state calculation by fast computing machines. J. of Chemical Physic, 21 :1087–1092.

[38] (https://docplayer.fr/2651733-Cours-des-methodes-de-resolution-exactes-heuristiques-et-metaheuristiques.html) (consulté le 14 juin 2021).

[39] F. Glover.(1986). Future paths for integer programming and links to artificial intelligence. Computers and Operations Research, 13 :533–549.

[40] F. Glover and M. Laguna.(1997). Tabu Search.

[41] (https://slideplayer.fr/slide/1752884/) (consulté le 16 juin 2021).

[42] J. Holland.(1975). Adptation in natural and aritificial systems. University of Michigan Press.

[43] (https://ledatascientist.com/algorithme-genetique/) (consulté le 16 juin 2021).

[44] C. COTTA, A. J. FERNANDEZ, J. E. GALLARDO, G. LUQUE, and E. ALBA .(2009).Metaheuristics in Bioinformatics: DNA Sequencing and Reconstruction ,Universidad de Malaga, Spain.

[45] C. Notredame and D. G. Higgins.(1996). SAGA: sequence alignment by genetic algorithm. Nucleic Acids Research, 24:1515–1524.

[46] C. Zhang and A. K. Wong.(1997). A genetic algorithm for multiple molecular sequence alignment. Bioinformatics, 13(6):565–581.

[47] C. Cotta.(2005). Memetic algorithms with partial Lamarckism for the shortest common super- sequence problem. In J. Mira and J. R. Alvarez, eds.(2005). ´ Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach, vol. 3562 of Lecture Notes in Computer Science. Springer-Verlag, New York, pp. 84–91.

[47] . Karpenko, J. Shi, and Y. Dai.(2005). Prediction of MHC class II binders using the ant colony search strategy. Artificial Intelligence in Medicine, 35(1–2):147–156.

[48] T. K. Rasmussen and T. Krink. (2003).Improved hidden Markov model training for multiple sequence alignment by a particle swarm optimization–evolutionary algorithm hybrid.Biosystems, 72(1–2):5–17.

[49] R. Parsons, S. Forrest, and C. Burks.(1995). Genetic algorithms, operators, and DNA frag- ment assembly. Machine Learning, 21:11–33.

[50] L. Li and S. Khuri. (2004).A comparison of DNA fragment assembly algorithms. In Pro- ceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences, pp. 329–335.

[51] E. Alba, G. Luque, and S. Khuri.(2005). Assembling DNA fragments with parallel algo- rithms. In B. McKay, ed.(2005). Proceedings of the Congress on Evolutionary Com- putation, Edinburgh, UK, pp. 57–65.

[52] P. Meksangsouy and N. Chaiyaratana. (2003).DNA fragment assembly using an ant colony system algorithm. In Proceedings of the Congress on Evolutionary Computation,vol. 3. IEEE Press, New York , pp. 1756–1763.

[53] A. Kel, A. Ptitsyn, V. Babenko, S. Meier-Ewert, and H. Lehrach.(1998). A genetic algorithm for designing gene family-specific oligonucleotide sets used for hybridization: the g protein-coupled receptor protein superfamily. Bioinformatics, 14:259–270.

[54] V. G. Levitsky and A. V. Katokhin. (2003).Recognition of eukaryotic promoters using a genetic algorithm based on iterative discriminant analysis. In Silico Biology, 3(1):81–87.

[55] K. Deb, S. Agarwal, A. Pratap, and T. Meyarivan. (2000).A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In Proceedings of the Parallel Problem Solving from Nature VI,pp. 849–858.

[56] C. Cotta, A. Mendes, V. Garcia, P. Franca, and P. Moscato. (2003).Applying memetic algo- rithms to the analysis of microarray data. In G. Raidl et al., eds.(2003). Applications of Evolutionary Computing, vol. 2611 of Lecture Notes in Computer Science. Springer- Verlag, New York, pp. 22–32.

[57] A. Mendes, C. Cotta, V. Garcia, P. Franc ̧a, and P. Moscato.(2005). Gene ordering in microarray data using parallel memetic algorithms. In T. Skie and C.-S. Yang, eds,Proceedings of the International Conference on Parallel Processing Workshops,Oslo, Norway. IEEE Press, New York, pp. 604–611.

[58] X. Xiao, E. R. Dow, R. Eberhart, Z. B. Miled, and R. J. Oppelt. (2003).Gene clustering using self-organizing maps and particle swarm optimization. In Proceedings of the 6th IEEE , International Workshop on High Performance Computational Biology, Nice, France, Apr.

[59] C. Notredame, L. Holm, and D. G. Higgins. (1998).COFFEE: an objective function for multiple sequence alignments. Bioinformatics, 14(5):407–422.

[60] G. Fogel and D. Corne.(2002). Evolutionary Computation in Bioinformatics. Morgan Kauf-mann, San Francisco, CA.

[61] C. Cotta.(2003). Protein structure prediction using evolutionary algorithms hybridized with backtracking. In J. Mira and J . R. Alvarez ,eds.(2003).,´ Artificial Neural Nets Problem Solving Methods, vol. 2687 of Lecture Notes in Computer Science. Springer-Verlag, New York, pp. 321–328.

[62] J. J. Gray, S. Moughon, C. Wang, O. Schueler-Furman, B. Kuhlman, C. A. Rohl and D. Baker. (2003).Protein–protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations. Journal of Molecular Biology, 331(1):281– 299.

[63] R. Santana, P. Larranaga, and J. A. Lozano. (2004).Protein folding in 2-dimensional lat- tices with estimation of distribution algorithms . In J . M Barreiro , F. Mart´ın- Sanchez,´V. Maojo, and F. Sanz, eds.(2004). Proceedings of the 5th Biological and Medical Data Analysis International Symposium, vol. 3337 of Lecture Notes in Computer Science. Springer-Verlag, New York,pp. 388–398.

[64] P. O. Lewis.(1998). A genetic algorithm for maximum likelihood phylogeny inference using nucleotide sequence data. Molecular Biology and Evolution, 15:277–283.

[65] C. Cotta and P. Moscato. (2002).Inferring phylogenetic trees using evolutionary algorithms. In J. J. Merelo et al., eds.(2002). Parallel Problem Solving from Nature VII, vol. 2439 of Lecture Notes in Computer Science. Springer-Verlag, New York, pp. 720– 729.

[66] C. Cotta.(2006). Scatter search with path relinking for phylogenetic inference. European Journal of Operational Research, 169(2):520–532.

[67] C. Cotta. (2006).Scatter search with path relinking for phylogenetic inference. European Journal of Operational Research, 169(2):520–532,.

[68] Zemin Ning, Anthony J. Cox and James C. Mullikin.(2001). SSAHA: A Fast Search Method for Large DNA Databases , Genome Research ,Cold Spring Harbor Laboratory Press ,11, 1725-1729 p .

[69] Homo sapiens GRCh38.p13 , refSEq NC_000001.11, https://www.ncbi.nlm.nih.gov/genome/?term=human%5Borganism%5D

Présenté par : HARCHAOUI Ikram Haifa
ABID Hasna

Date de Soutenance : 08/07/2021

Intitulé : Apport des métaheuristique pour la recherche de sous séquences nucléiques ou protéiques

Résumé :

La recherche de séquence permet à un utilisateur de rechercher une sous-séquence d'ADN spécifique dans une séquence d'ADN ou une base de données plus grande. Dans ce contexte, les algorithmes de recherche de séquence rapide joueront un rôle important dans l'exploitation des informations contenues dans les données nouvellement séquencées. Les métaheuristiques sont des méthodes approchées et sont des procédures de haut niveau conçues pour résoudre des problèmes d'optimisation. Les métaheuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global, c'est-à-dire l'extremum global d'une fonction, par échantillonnage d'une fonction objectif. L'objectif de notre mémoire c'est d'implémenté un programme avec le langage python basé sur la méthode approcher la particule swarm optimisation (PSO) pour le but de trouvé la sous-séquence la plus longue et le comparé avec une autre méthode exacte la programmation dynamique (DP). Et obtenir un meilleur résultat.

Mots clés : Métaheuristique, optimisation, particule swarm optimisation, recherche de sous séquence.

❖ **Jury d'évaluation :**

- Président de jury : Dr. BELLIL Inès — MCA. UFM. Constantine 1
- Encadreur : Dr. DAAS Mohamed Skander — MCB. UFM. Constantine 1.
- Examinateur : Dr. GHERBOUDJ Amira — MCA. UFM. Constantine 1

Année universitaire : 2020/2021