REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : 22/M2/2021

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Systèmes d'information, Optimisation et Décision (SIOD)**

Un algorithme hybride pour le problème d'assemblage de fragments d'ADN

Par :
KHALIL ZIANI

Soutenu le 28/06/2022 devant le jury composé de :

Bougtuetitiche Amina	MCB	Président
Moussaoui Manel	MAA	Rapporteur
Belouannar Saliha	MAA	Examinateur

Année universitaire 2021-2022

Résumé

Le problème d'assemblage de fragments consiste à construire une séquence d'ADN à partir de plusieurs centaines (voire des milliers) de fragments obtenus par les biologistes du laboratoire. Pour résoudre ce problème, nous avons adopté l'algorithme PALS (Problem Aware Local search). Pour améliorer la recherche de bonne solution, Notre idée principale du intégré lalgorithme génétique (AG) comme un opérateur de recherche additionnel dans l'algorithme PALS.

Mots-clés : ADN, PALS, AG, Cutoff, Contigs, Fitness

ملخص:

تتكون مشكلة تجميع الشظايا من بناء تسلسل DNAمن عدة مئات (أو حتى آلاف)من الأجزاء التي حصل عليها علماء الأحياء في المختبر .لحل هذه المشكلة ، اعتمدنا خوارزمية PALS(البحث المحلي عن المشكلة).لتحسين البحث عن حل جيد ، تتمثل فكرتنا الرئيسية في دمج الخوارزمية الجينية (GA)كعامل بحث إضافي في خوارزميةPALS .

الكلمات المفتاحية : ADN, PALS, AG, Cutoff, Contigs, Fitness

Remerciements

*Tout d'abord, je remercie **"Dieu"** Tout Puissant de m'avoir donné la force et la patience nécessaire pour achever ce travail de mémoire.*

*J'ai tenon d'abord à remercier mon encadreur **"Moussaoui Manel"** de sa disponibilité, son soutien continu, sa motivation qui a fortement contribué à mener à bien ce travail.*

Je remercie également les membres de jury, qui vont accepter de lire et d'évaluer ce travail.

Mon remerciement s'étend également à tous nos enseignants pendant les années des études, aussi à l'équipe de département d'informatique.

Enfin, je tiens à remercier tous ceux qui m'ont aidé en particulier ma famille.

Dédicaces

Je dédie cet humble travail :

À mes très chers parents qui ont fait de moi ce que je suis aujourd'hui, que ce travail traduit ma gratitude et mon affection

À toute ma famille, source d'espoir et de motivation

À mes frères Abd Elmalek, Abd Elaziz

À mes surs Khededja, Hadil, Nor Elhoda, Nawal

À mes collègues Safiya Intisar Rania ou Rania Safiya Intisar, je vous remercie de votre patience avec moi. Merci d'être si proche de moi. Le premier cadeau était de Rania après l'obtention du diplôme. Merci mon ami.

Mes amis avec qui je souris

À tous mes amis Chouaib Azzouz, Assil Lehmaidi, Abdallah Chiba

Table des matières

Table des matières vi

Table des figures viii

Liste des tableaux x

Introduction générale 1

1 Problème d'assemblage de fragments d'ADN 3

1.1 Introduction . . . 3

1.2 Analyse de l'ADN . . . 3

1.2.1 Structure de l'ADN . . . 5

1.2.2 Les étapes de l'analyse de l'ADN . . . 6

1.2.2.1 Réplication de la séquence d'ADN . . . 6

1.2.2.2 Fragmentation de l'ADN . . . 7

1.3 Séquençage de fragments d'ADN . . . 7

1.3.1 Les méthodes de séquençage . . . 8

1.3.1.1 Séquençage de première génération . . . 8

1.3.1.2 Séquençage de deuxième génération (NSG) . . . 9

1.3.1.3 Les nouvelles technologies de séquençage à très haut débit (NGST) . . . 9

1.3.2 Le séquençage de Shotgun . . . 10

1.4 L'assemblage de fragments d'ADN . . . 11

1.4.1 Assemblage par cartographie (Mapping) . . . 11

1.4.2 Assemblage de novo . . . 11

1.4.3 L'approche OLC (Overlap-Layout-Consensus) . . . 12

1.5 Les problèmes d'assemblage....13
1.6 Conclusion....14

2 Algorithme de recherche local PALS et Algorithme génétique 15

2.1 Introduction....15
2.2 Méthodes heuristiques....15
2.3 Méthodes méta heuristiques....16
2.4 Problème NP....16
2.5 Classification des métas heuristiques....17
2.6 Principe des méthodes méta heuristique les plus répondues....18
2.6.1 Voisinage....18
2.6.2 Recherche locale....19
2.6.2.1 Principe de recherche locale....20
2.6.2.2 L'algorithme PALS....20
2.6.2.3 L'algorithme génétique (AG)....23
2.6.2.4 PALS avec l'algorithme Génétique....31
2.7 Conclusion....31

3 Conception et implémentation 32

3.1 Introduction....32
3.2 Description global et détails du système....32
3.3 Environnement de développement....33
3.3.1 MATLAB....33
3.3.2 Base de données utilisée....34
3.3.3 Généré une solution initiale....37
3.3.4 Calculer le score de chevauchement Δ_f et le score de contigs Δ_c : . 38
3.3.4.1 Calculer de Fitness....38
3.3.4.2 Calculer le nombre de contigs....40
3.3.5 Mémoriser les mouvements dans une liste....41
3.4 Sélectionner un mouvement par (AG)....41
3.4.1 Génération de la population initiale....41
3.4.2 Evaluation de la population....42
3.4.3 Sélection....42

3.4.4 Croisement.....42
3.4.5 Mutation..... 43
3.4.6 Sélectionner le meilleur mouvement..... 44
3.4.7 Critère d'arrêt..... 44
3.5 Expérimentation et Résultats.....44
3.6 Conclusion.....45

Conclusion générale **45**

Bibliographie **46**

Table des figures

1.1 Origine d'ADN. [3] 4
1.2 Structure de l'ADN. [5] 5
1.3 Structure d'un nucléotide d'ADN.[5]. 5
1.4 Processus d'analyse d'ADN. [8] 6
1.5 Duplication de l'ADN. [11] 7
1.6 Un séquenceur d'Illumina. [15] 8
1.7 Séquençage de Sotgun. [20]..... 10
1.8 Etape de chevauchement (overlap)..... 13

2.1 Classification des métas heuristiques. [26].....18
2.2 Procédure générale de la recherche locale. [2]..... 19
2.3 Evolution d'une solution dans la méthode de recherche locale. [29]..... 20
2.4 Algorithme : PALS. [21]..... 21
2.5 Algorithme : Fonction CalculateDelta [21].....23
2.6 Processus d'un algorithme génétique. [31]..... 25
2.7 Représentation schématique du croisement CX. [8].....28
2.8 Représentation schématique de mutation. [8].....29
2.9 Représentation schématique de la mutation par inversion.....29
2.10 Représentation schématique de la mutation par déplacement. [8].....30
2.11 Représentation schématique de la mutation par permutation. [8]..... 30

3.1 Description global du système.....33
3.2 Description détails du système 33

3.3 MATLAB.....33
3.4 Liste de fragments..... 36
3.5 Chevauchement entre les fragments.....37

3.6 La solution initiale..38

3.7 Sélection d’une population initiale.....42
3.8 Croisement (CX)..... 43
3.9 Mutation..... 43
3.10 Le meilleur mouvement..... 44
3.11 La solution finale.....44

Liste des tableaux

3.1 Les caractéristiques de Matériel..... 34
3.2 Base de données utilisée GenFrag..... 34
3.3 Comparaison entre PALS vs PALS avec algorithme génétique.....45

Introduction général

Le problème d'assemblage de fragments d'ADN est un problème issu du domaine de la Bioinformatique.Ce problème est un problème de rechercher un ordre totaldes fragments donnés qui aboutit à une séquence consensus reflétant avec précision la séquencé parent.

Le séquençage d'ADN est devenu une pratique indispensable dans les domaines de médecine, de phylogénétique et du crime. Il consiste à construire une séquence d'ADN à partir de plusieurs centaines (voire des milliers) de fragments obtenus par les biologistes du laboratoire.

L'émergence de ce problème dans la bioinformatique a été la conséquence du besoin d'automatiser le séquençage de séquence de l'ADN de taille considérable ce problème classécomme un problème NP-Difficile [Pevener 2006] pour n fragments, il y a n ! Configurations possibles. [1]

L'algorithme PALS (de l'anglais Problème Aware Local Search) est une méthode de recherche locale plus rapide et plus compétitif, développe par Alba et Luque (Alba et Luque, 2007) pour résoudre ce problème. [1]

Deux problèmes major dans l'algorithme PALS : premièrement comment générer la solution de départ (generate Initial Solution), et deuxièmement comment sélectionner un mouvement parmi une liste de mouvement à chaque itération (select Movement).

Dans ce travail nous avons utilisé l'algorithme génétique pour résoudre ces deux problèmes : utiliser AG pour sélectionner le meilleur mouvement dans l'algorithme PALS et appliquer la solution obtenue par AG pour réinitialiser PALS.

Ce mémoire débute par une introduction générale qui présente la problématique. Avec trois chapitres :

- Le premier chapitre présente le problème d'assemblage de fragments ADN et les stratégies de séquençage et les approches d'assemblage.
- Le deuxième chapitre présente la recherche locale, l'algorithme PALS et les algorithmes

génétique.

- La conception et l'implémentation du l'algorithme, les expérimentations et les résultats sont abordés dans le troisièmechapitre.

Chapitre 1

Problème d'assemblage de fragmentsd'ADN

1.1 Introduction

Le problème d'assemblage de fragments d'ADN (ADN FAP) consiste à construire une séquence d'ADN à partir de centaines (ou de milliers) de fragments obtenus par des biologistes de laboratoire. C'est un problème du domaine de la bio-informatique et c'est un enjeu important dans tout projet de génome. Ce chapitre aborde le problème de l'assemblage de fragments d'ADN qui se produisent au niveau du séquençage génomique. Nous allons commencer par le concept d'analyse de l'ADN, puis le processus de séquençage de l'ADN, et quelques techniques d'assemblage.

1.2 Analyse de l'ADN

L'ADN, qui signifie acide désoxyribonucléique, est une molécule qui contient le code génétique humain. Cette information génétique est présente dans chaque cellule du corps humain. Le code génétique correspond à la séquence de 4 nucléotides ACGT : adénosine, cytidine, guanosine et thymidine. [2]

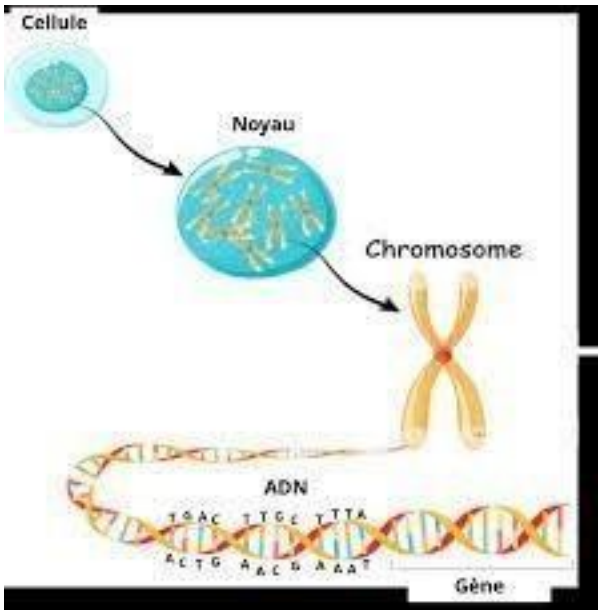


FIGURE 1.1 – Origine d'ADN. [3]

Le but principal de l'analyse de l'ADN est de déterminer le séquençage complet du génome et de déterminer son contenu génétique. À l'heure actuelle, l'analyse de l'ADN est devenue une pratique incontournable dans les domaines de la médecine, de la phylogénétique et de la médecine légale (police technique), comme l'indique le paragraphe suivant :
Les preuves ADN ont provoqué des bouleversements majeurs dans les communautés scientifiques et juridiques, rendant l'identification génétique possible par des moyens autres que les tests sanguins conventionnels. Les méthodes d'analyse de l'ADN ont été améliorées au paragraphe 5 du chapitre 01, Assemblage des fragments d'ADN, afin que des résultats concluants puissent être obtenus à partir d'un minuscule échantillon d'ADN. Pour démontrer le haut degré de fiabilité de leurs méthodes, les laboratoires qui effectuent des examens médico-légaux sont accrédités. L'un des avantages de l'utilisation de preuves ADN est la possibilité d'établir plus précisément les liens familiaux entre personnes apparentées. [4] Dans ce cas, l'analyse d'ADN permet de :

- Identifier des victimes, d'établir la présence de suspects sur le lieu du crime.
- Etablir des relations fonctionnelles entre les virus, les symptômes et les maladies, etc.
- Etudier les relations évolutives des espèces dans l'analyse phylogénétique.
- Classifier des espèces, en biologie.

1.2.1 Structure de l'ADN

L'ADN (pour acide désoxyribonucléique) est le support de l'hérédité, sa structure a été découverte en 1953 par Watson et Crick. Les molécules d'ADN sont les plus grosses molécules du monde vivant, elles sont présentées dans tous les organismes vivants. Une molécule d'ADN est une double hélice composée de deux brins enroulés l'un autour de l'autre, chacun de ces brins est constitué d'assemblage de plusieurs nucléotides accrochés les uns aux autres par des liaisons phosphodiester, dont l'ordre d'enchaînement est très précis et correspond à l'information génétique. [5]

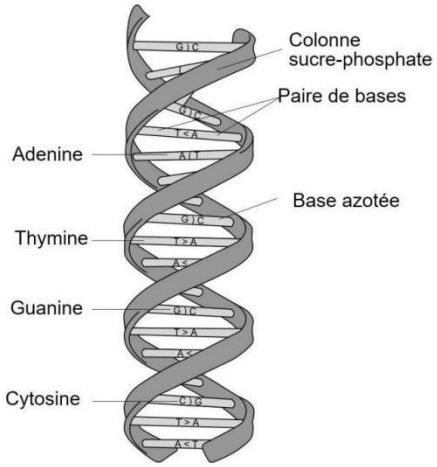


FIGURE 1.2 – Structure de l'ADN. [5]

Un nucléotide est constitué un sucre (désoxyribose), d'un acide phosphorique et d'une base azotée. 4 bases sont possibles :

· Adénine. • Thymine. • Guanine. • Cytosine. [6]

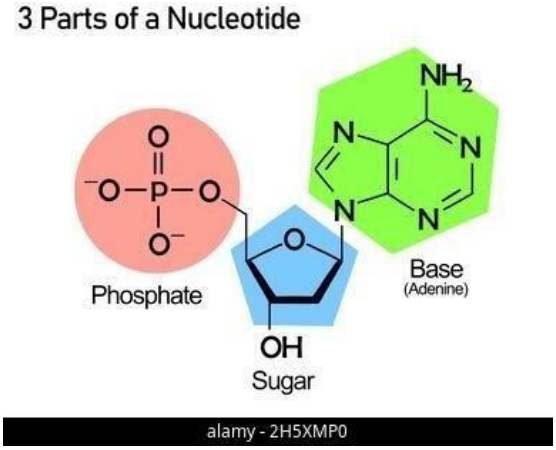


FIGURE 1.3 – Structure d'un nucléotide d'ADN. [5]

1.2.2 Les étapes de l'analyse de l'ADN

L'extraction de l'ADN se fait à partir de la totalité ou d'une partie de l'organisme étudié, même si un grand nombre de cellules est utilisé pour réaliser l'extraction, la quantité finale d'ADN est généralement faible. Alors l'étape suivante consistera donc à amplifier cette quantité de l'ADN extraite, puisque la séquence de l'ADN est très longue (la taille du génome humain est approximativement de 3,2 milliards de paires de nucléotides), on doit la découper pour l'analyser. La figure (1.4) montre le processus d'analyse d'ADN. [7]

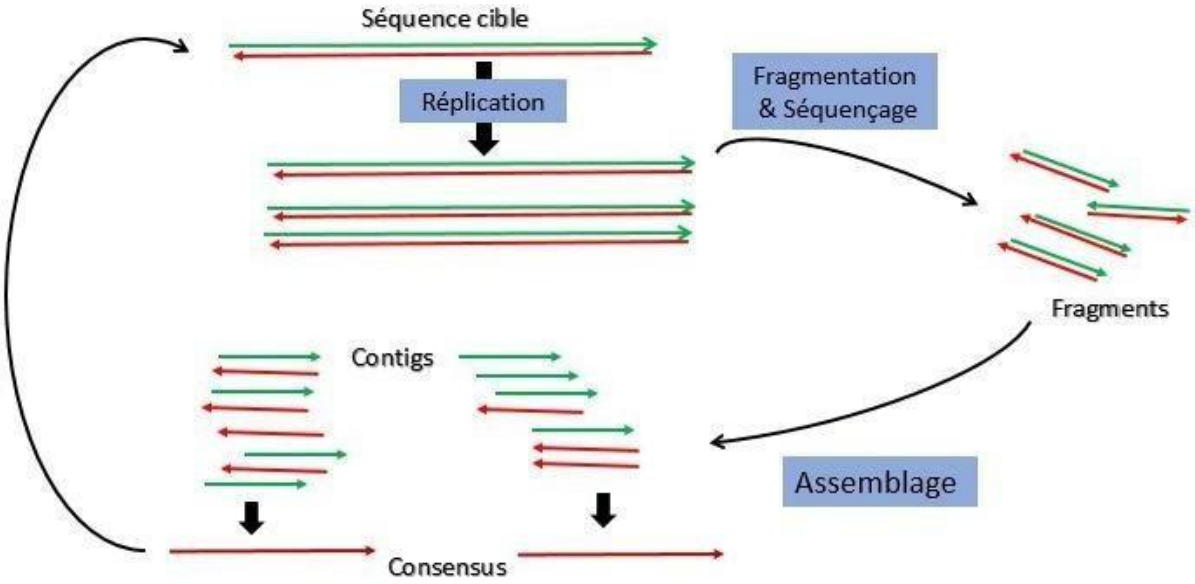


FIGURE 1.4 – Processus d'analyse d'ADN. [8]

1.2.2.1 Réplication de la séquence d'ADN

La réplication de la séquence d'ADN est l'opération de duplication ou d'amplification, elle est réalisée à partir d'une technique appelée PCR ou "Polymerase Chain Reaction" (réaction de polymérisation en chaîne). Cela permet de dupliquer à plusieurs millions d'exemplaires un fragment d'ADN grâce à l'ADN polymérase. [9]

Une duplication est une mutation qui génère un dédoublement d'une partie de l'ADN génomique. La duplication peut recouvrir l'ensemble du génome (formation de polyploïdes), un chromosome entier, ou un fragment de chromosome de taille plus ou moins grande. Les duplications peuvent éventuellement entraîner l'apparition de copies multiples d'un ou plusieurs gènes, provoquant ainsi une certaine redondance de l'information génétique. [10]

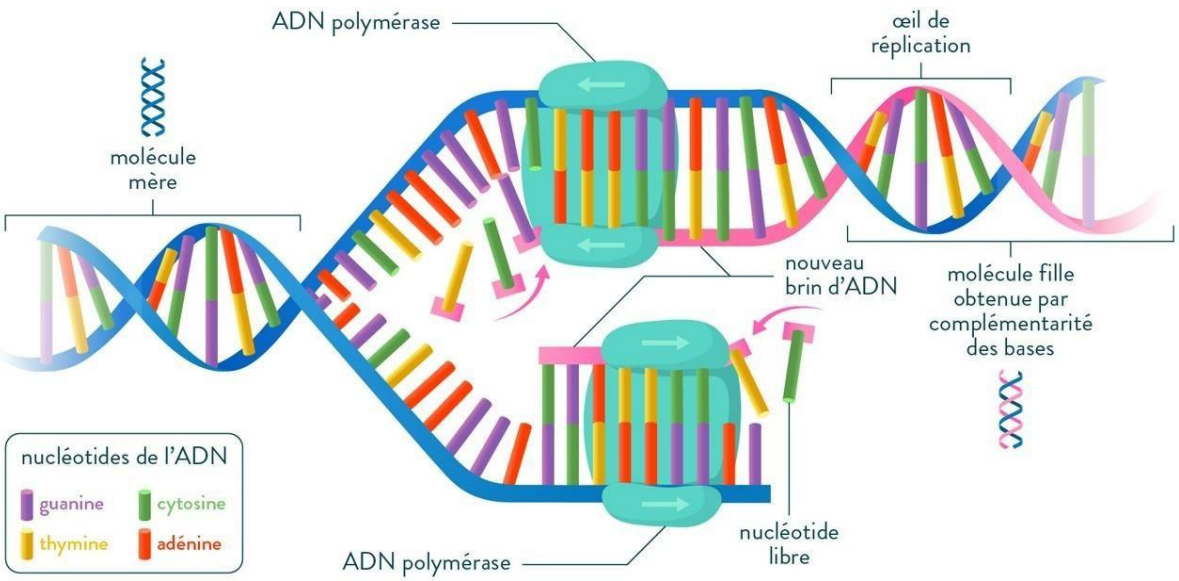


FIGURE 1.5 – Duplication de l'ADN. [11]

1.2.2.2 Fragmentation de l'ADN

L'analyse d'une molécule d'un ADN de taille importante nécessite son découpage en fragments de taille compatible avec le séquençage. Deux différentes approches de fragmentation ont principalement été utilisées :

- **La fragmentation aléatoire :** on découpe directement l'ensemble de l'ADN à séquencer en petits morceaux de taille optimisée pour le séquençage (1000 paires de bases). [12]
- **Segmentation après cartographie :**dans le cas des génomes de très grande taille, la complexité de reconstruction devient une difficulté sérieuse, c'est pourquoi dans ces cas-là, certaines équipes ont eu recours à une approche à deux niveaux. [13]

1.3 Séquençage de fragments d'ADN

La première étape dans l'étude du génome d'un organisme spécifique est la détermination de la séquence complète d'ADN : c'est le processus de séquençage d'ADN. Les informations fournies par cette étape servent à identifier les fonctions des gènes dans le génome.

Le séquençage d'ADN est le processus qui consiste à déterminer l'ordre d'enchaînement des nucléotides d'un fragment d'ADN donné. Il est réalisé par des machines de séquençage (des séquenceurs). [14]

Un séquenceur de gènes est un appareil capable d'automatiser l'opération de séquençage de l'ADN. Il peut déterminer la séquence des bases A, C, T et G, de gauche à droite le long d'un brin d'une double hélice d'ADN jusqu'à une longueur moyenne typique, et a un taux d'erreur typique. La courte séquence fournie par le séquenceur est appelé une lecture (issue du terme anglais read). La figure (1.6) montre des exemples de séquenceurs d'Illumina.

Les capacités d'un séquenceur sont définies par :

- La longueur des lectures (reads) produits (L).
- Le nombre des lectures (reads) produits (n).
- Le nombre de nucléotides lu : (L x n).
- Le temps de séquençage.
- La qualité du séquençage.

FIGURE 1.6 – Un séquenceur d'Illumina. [15]

1.3.1 Les méthodes de séquençage

Apres l'apparition de la technique de séquençage, différentes méthodes de séquençage de l'ADN ont été développées

1.3.1.1 Séquençage de première génération

Le séquençage de l'ADN a été inventé dans la deuxième moitié des années 1970. Deux méthodes ont été développées indépendamment en 1977, l'une par l'équipe de Maxam et Gilbert aux États-Unis, et l'autre par Frederick Sanger en Grande-Bretagne. Ces deux méthodes sont fondées sur des principes diamétralement opposés ; l'approche de Sanger

estune méthode par synthèse enzymatique sélective, tandis que celle de Maxam et Gilbert est une méthode par dégradation chimique sélective.

A la fin des années 80, une automatisation de la méthode Sanger 80 a été réalisée, avec le développement des marquages fluorescents et de l'électrophorèse capillaire, qui a ouvert la voie du séquençage à haut débit.

La technique de séquençage avec des didésoxyribonucléotides (ddNTP) fluorescents ("dye terminator sequencing") utilise des didésoxyribonucléotides dont chacun est marqué par un fluorophore spécifique. Les fragments d'ADN synthétisés portent ce fluorophore terminal. On les appelle des terminateurs d'élongation ou "Big Dye Terminators" ou "Dye-labeled terminator". [16]

1.3.1.2 Séquençage de deuxième génération (NSG)

Le terme NGS ń Next generation sequencing ż regroupe l'ensemble des technologies ou plateformes de séquençage développées depuis 2005 par quelques sociétés de biotechnologies, l'émergence de cette nouvelle technologie est considérée comme la seconde génération de séquençage. [17]

Avec ces technologies, on peut séquencer de grandes quantités d'ADN en des temps records, et la taille d'un fragment peut être entre 150 à 300 pb.

1.3.1.3 Les nouvelles technologies de séquençage à très haut débit (NGST)

Les améliorations technologiques ont rapidement changé le domaine du séquençage de l'ADN, une révolution en génomique fonctionnelle a eu lieu depuis 2012, avec l'avènement des technologies de séquençage à très haut débit NGST ("next-génération high throughput DNA sequencing technologies")[18].

Des quelques 800 à 1000 nucléotides qu'un chercheur pouvait espérer séquencer en quelques jours par des techniques lourdes, complexes et dangereuses (utilisation d'isotopes radioactifs) dans les années 80, on est arrivé à l'heure actuelle à des techniques de séquençage simplifiées qui génèrent des millions de nucléotides par jour. L'ensemble des données de séquençage est implémenté en temps réel dans des bases de données pour leur analyse.

1.3.2 Le séquençage de Shotgun

Le séquençage de Shotgun est méthode de séquençage introduite en 1982 par Sanger. Cette methode commence par découper de façon aléatoire le génome en différentes sections de longueur prédéterminée : 2.000 paires de base, 10.000 paires de base ou 50.000 paires de base.

Des algorithmes mathématiques permettent ensuite d'assembler les fragments qui se suivent et de leur attribuer leur véritable emplacement sur le génome [19].

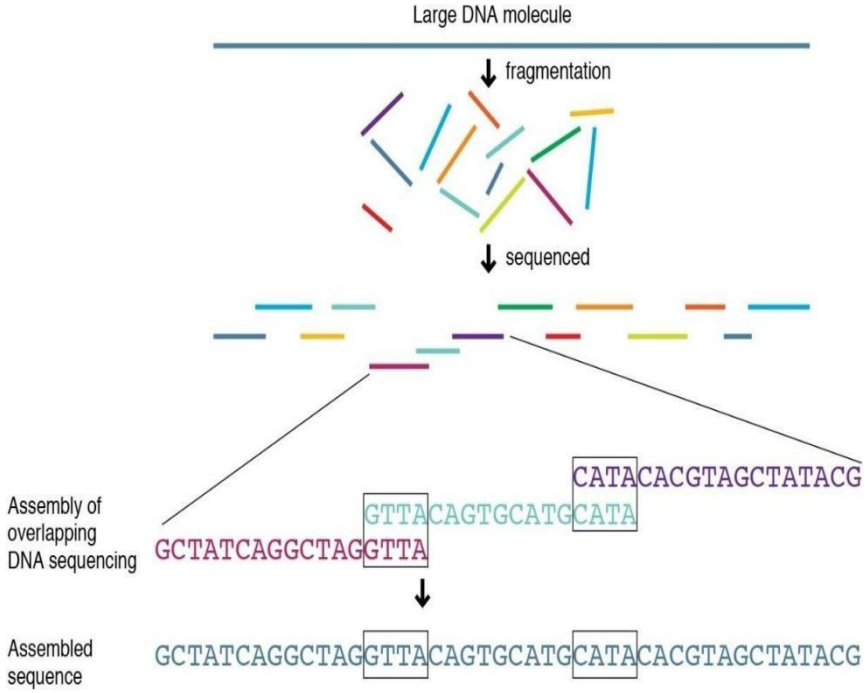


FIGURE 1.7 – Séquençage de Sotgun. [20]

La méthode de séquençage de Sotgun comporte les étapes suivantes :

- Extraire l'ADN : L'extraction de l'ADN se fait à partir de la totalité ou d'une partie de l'organisme étudié. De nombreux protocoles existent.
- Découpage du génome en fragments qui sont insérés dans un vecteur (BAC, YAC,).
- Clonage de l'ADN et séquençage des extrémités du clone.
- Cosmide ou BAC sous-cloné en fragments de petite taille.
- Des clones prélevés aléatoirement sont séquencés.
- Recueil des données brutes du séquenceur.

- Des séquences contigües sont reconstruites par recouvrement.
- Assembler la séquence par création d'un graphe de DE Bruijn qui peut comporter plus de 100 millions de noeds et dont les segment sont appelées K-mers.
- Finalisation de la séquence en comblant les tous : une couverture plus importante aboutit à des trous plus petits et moins nombreux.
- Recherche de la signification phénotype et clinique via les polumorphisme vucléotides, insertion/suppression variant, variation du nombre d'exemplaire et mutations.

L'assemblage de fragments d'ADN consiste à reconstituer (ou bien fusionner) des fragments d'ADN issus d'une plus longue séquence et produites par un séquenceur.

1.4 L'assemblage de fragments d'ADN

Le programme informatique qui réalise cette tâche est appelé assembleur. Quelques termes sont employés lorsqu'on traite le problème d'assemblage de fragments d'ADN [21], comme :

— **Lecture (read) :** une séquence d'un fragment.

— **Contigs :** séquences continues générées par l'alignement de séquences de fragments qui se chevauchent.

— **Brèches ("gaps") :** parties du génome non séquencées ou dont les séquences ne chevauchent pas avec d'autres et ne peuvent donc entrer dans un contig.

— **Régions de faible complexité :** parties du génome dont les séquences sont très peu diversifiées comme les séquences répétées.

Il existe deux types d'assemblage :

1.4.1 Assemblage par cartographie (Mapping)

Son principe est de comparer les fragments issus de la phase de séquençage avec une séquence ADN déjà connue. Trouver une solution à ce problème revient donc à chercher, pour chacun des fragments la position qui marque un chevauchement maximal avec le génome de référence [21].

1.4.2 Assemblage de novo

L'assemblage de fragments ADN sans aucun génome de référence est appelé assemblage de novo. Il est utile dans le cas où l'ADN provient d'une source inconnue (par exemple les

cheveux trouvés sur une scène de crime). Ce type d'assemblage se base sur la vérification des chevauchements deux à deux entre les fragments. Trouver la solution exacte est équivalent à effectuer un nombre de comparaison exponentiel par rapport au nombre de fragments. Ceci est impossible de l'effectuer dans un temps raisonnable.

Il existe un certain nombre des méthodes d'assemblage de novo, l'approche OLC (Overlap Layout-Concensus) est l'une des premières méthodes utilisées pour assembler une séquence génomique avec succès.

1.4.3 L'approche OLC (Overlap-Layout-Consensus)

Le principe de l'approche OLC (Overlap-Layout-Consensus) : est de fusionner des séquences chevauchantes, en commençant par celles ayant les scores les plus élevés, jusqu'à obtention d'une séquence unique [21].

Les assembleurs utilisant la méthode ń overlap-layout-consensus ż sont plutôt utilisés avec des lectures allant de 100 à 800 pb principalement avec un séquençage Sanger. Le processus d'assemblage l'approche OLC se déroule en trois phases ou étapes. [21]

Chevauchement (overlap) : Cette phase consiste à trouver les fragments qui se chevauchent. Elle consiste à chercher le bon ou le plus long alignement entre le suffixe d'une lecture et le préfixe d'une autre lecture. L'intuition derrière la recherche des chevauchements entre paires de lectures est que deux lectures suffisamment chevauchantes se retrouvent très probablement l'une à côté de l'autre dans la séquence cible.

Alignement : (layout) Cette phase consiste à chercher un ordre plausible de fragments en se basant sur les chevauchements calculés. C'est la plus difficile étape parce que la décision d'assembler deux fragments se base sur leur chevauchement qui peut être approximatif du fait des erreurs de séquençage.

Pour la plupart des assembleurs, des paires de lectures sont sélectionnées en fonction de leur score (les scores les plus élevés en premier), si l'alignement est considéré comme cohérent, les deux séquences sont fusionnées pour former un contig. [21]

Consensus : Cette phase consiste à produire une séquence d'ADN à partir du résultat de la phase d'alignement.

Un exemple illustratif des trois phases de l'approche OLC est donné dans la figure (1.8).

- Etape de chercher les chevauchements.
- Etape d'alignement.

- Etape de Consensus.

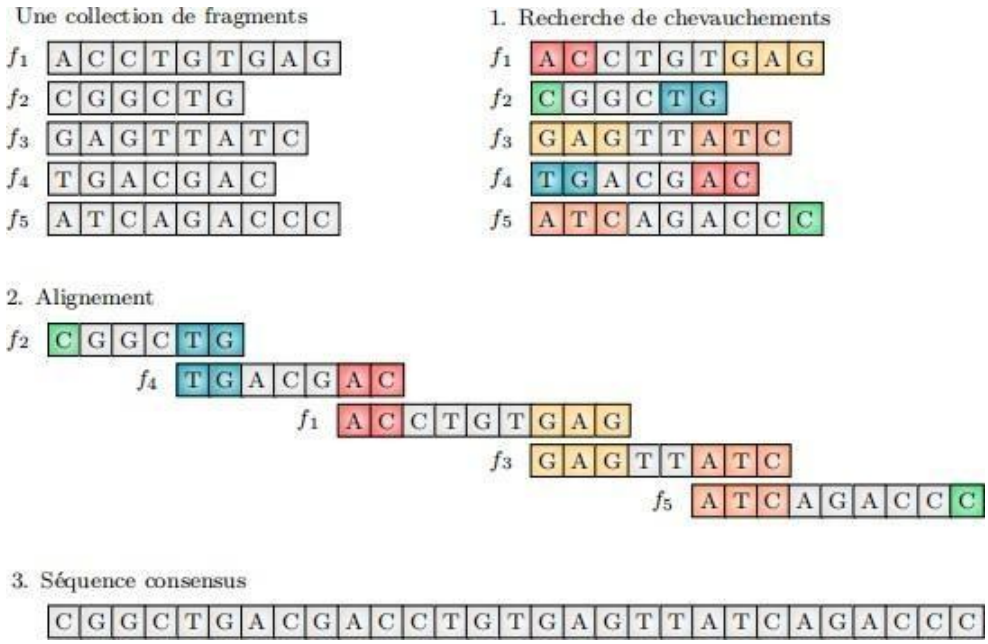


FIGURE 1.8 – Etape de chevauchement (overlap).

1.5 Les problèmes d'assemblage

Le problème de l'assemblage peut être comparé à celui de la reconstruction du texte d'un livre à partir de plusieurs copies de celui-ci, préalablement déchiquetées en petits morceaux. Parmi les problèmes d'assemblage d'ADN :

- Toutes les machines de séquençage imposent une longueur de lecture maximale et produisent des erreurs.
- Orientation inconnue : une fois la séquence d'ADN d'origine est fragmentée, l'orientation des fragments est perdue et l'extrémité à sélectionner devient inconnue.
- Régions répétées : pour un assembleur deux lectures se chevauchant parfaitement proviennent de la même région génomique. Or, certaines séquences peuvent se retrouver de multiples fois dans un génome.
- Les assembleurs actuels ne gèrent pas d'une manière assez efficace les longues séquences répétées.
- Faible couverture ; la distribution de la couverture sert à mesurer la qualité de la séquence consensus. Plus la couverture est grande, moins il y a de gaps, et meilleur est le résultat.

La couverture C est donnée par :

$$C = \quad | f | f \in L G$$

Ou :

- L est la collection de fragments d'ADN.
- $| f |$ est la longueur du fragment $f \in L$.
- G est la longueur de la séquence cible.

1.6 Conclusion

Dans ce chapitre, nous avons étudié expliqué le problème de l'assemblage de fragments d'ADN. Nous présentons quelques méthodes pour résoudre ce problème qui se distinguent par la caractérisation des génomes séquencés, le développement de techniques de séquençage et l'utilité du séquençage des génomes spécifiques. Nous avons vu que le processus d'assemblage fait face à de nombreuses difficultés qui rendent difficile le problème de l'assemblage de fragments d'ADN, donc l'inférence et la méta-science ont été proposées. Dans le chapitre suivant, nous présenterons les heuristiques et méta-heuristiques que nous utiliserons dans notre projet pour résoudre ce problème.

Chapitre 2

Algorithme de recherche local PALS et Algorithme génétique

2.1 Introduction

Dans ce chapitre nous allons parler sur les heuristiques qui sont des méthodes de résolution et stratégies de bon sens purement algorithmiques permettant d'obtenir des solutions à n'importe quel problème décisionnel rapidement, Ensuite, nous présentons une méta-heuristique ainsi que quelques notions fondamentales comme la recherche locale.

L'approche de la recherche locale est très efficace pour résoudre les problèmes NP-difficile ou NP-complet surtout pour ceux ayant un espace de recherche avec un voisinage de très grande taille, elle garantit une solution en temps abordable grâce à sa rapidité et à son principe relativement simple. Dans certain cas, la qualité de la solution trouvée n'est pas bonne. [2]

Ce chapitre est consacré à la notion de la recherche locale. Nous ferons en particulier une description de l'algorithme PALS.

2.2 Méthodes heuristiques

Une heuristique est une méthode, conçue pour un problème d'optimisation donné, qui produit une solution non nécessairement optimale lorsqu'on lui fournit une instance de ce problème.

Les méthodes heuristiques sont utilisées pour des problèmes de grandes tailles pour lesquels les méthodes exactes prennent un temps exponentiel. Généralement, on n'obtient

pas la solution optimale mais une solution approchée, assez bonne. Une méthode heuristique se déplace dans l'espace de solutions, de voisin en voisin, en se basant seulement sur les informations locales (la solution courante et son voisinage), avec objectif d'atteindre un optimum au-delà duquel aucun mouvement local n'est possible. Cette méthode de par sa simplicité est caractérisée par une moindre consommation de mémoire, mais elle ne dirige pas généralement la résolution du problème vers un optimum global. [21]

Bien que le principe soit assez générique, il doit cependant être adapté en fonction du problème, car ces heuristiques sont généralement guidées par des caractéristiques spécifiques au problème considéré et en sont donc totalement dépendantes. [22]

2.3 Méthodes méta heuristiques

Les métas heuristiques sont des heuristiques adaptées à chaque problème, qui ont été misent au point, sans changements majeurs dans l'algorithme, afin de résoudre des problèmes d'optimisation difficile, pour lesquels nous ne connaissons pas de méthodes classiques plus efficaces. Ces méthodes sont, pour la plupart, inspirées de la biologie (algorithmes évolutionnaires) ou de l'éthologie (essaims particulaires, colonies de fourmis). [23]

Un méta heuristique est formellement défini comme un processus de génération itérative qui guide un heuristique subordonné en combinant des concepts intelligemment différents pour explorer et exploiter l'espace de recherche, des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver des solutions efficaces presque optimales. [24]

Les méta-heuristiques sont des méthodes généralisées pour la résolution de problèmes NP. Ainsi, à l'inverse des heuristiques, elles sont indépendantes du problème considéré. De plus, des preuves de convergence vers la solution optimale ont été établies (FAIGLE et KERN [1992]) montrant que la probabilité de trouver la solution optimale augmente si on laisse un temps infini. Bien qu'en pratique il n'est pas possible de laisser un temps infini, les méta-heuristiques ont montré de très bonnes performances sur un temps fini. [22]

2.4 Problème NP

Est un problème connu comme étant NP et si une solution au problème est connue, la démonstration de l'exactitude de la solution peut toujours être réduite à une vérification P

(temps polynomial). Si P et NP ne sont pas équivalents, la solution des problèmes de NP nécessite (dans le pire des cas) une recherche exhaustive. [25]

Soit avec une méthode exacte pour chercher l'optimum global, soit avec une méthode approchée pour chercher une solution de bonne qualité dans un délai raisonnable. Une résolution avec une méthode exacte pour un problème NP-difficile est souvent couteuse car le temps d'exécution est non polynomial en la taille des instances à traiter. Si les instances du problème sont de petites tailles, la résolution avec une méthode exacte reste envisageable. Par contre, si les instances sont de grandes tailles, une méthode de résolution approchée reste l'unique façon pratique de résoudre le problème. Le principe d'une telle approche est une exploration de l'espace des solutions à la recherche d'une bonne solution (pas forcement l'optimale) en un temps d'exécution raisonnable. [26]

Un problème est attribué à la classe NP (temps polynomial non déterministe) si elle est résolue en temps polynomial par une machine de Turing non déterministe. [25]

2.5 Classification des métas heuristiques

Il existe de nombreuses façons pour classer les métas heuristiques. Ce diagramme tente de présenter où se placent quelques-unes des méthodes les plus connues. Un élément présenté à cheval sur différentes catégories indique que l'algorithme peut être placé dans l'une ou l'autre classe, selon le point de vue adopté. [27]

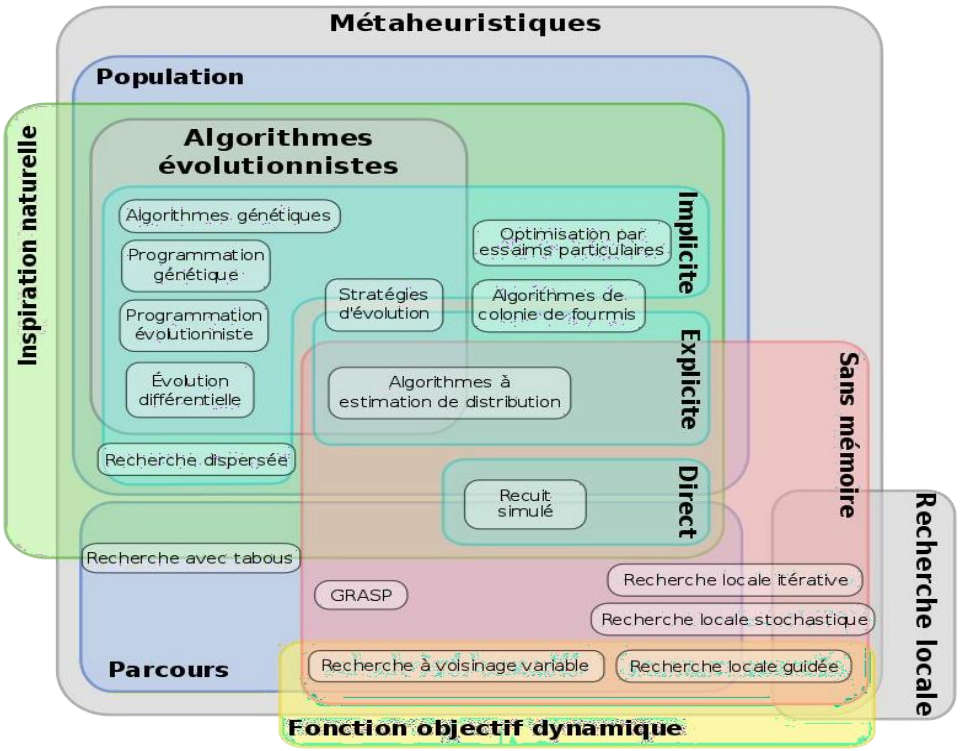


FIGURE 2.1 – Classification des métas heuristiques. [26]

2.6 Principe des méthodes méta heuristique les plus répondues

2.6.1 Voisinage

Le principe général le plus largement utilisé dans l'élaboration des méta heuristiques est celui de voisinage. À chaque solution s du problème, on associe un sous ensemble V(s) de solutions. Ce sous-ensemble peut être statique, comme dans le cas du recuit simulé, mais aussi dynamique et dépend du temps t (ou plus précisément de l'itération à laquelle on se trouve).

Pour un problème d'optimisation combinatoire non convexe, pour lequel il est possible de définir un ensemble de voisinages a priori intéressants, la situation se complexifie. Il devient alors difficile de se décider pour l'un ou l'autre des voisinages autrement que par des essais. Comme le voisinage n'est qu'une partie des principes, tous interdépendants, utilisés dans l'heuristique, le choix d'un (ou de plusieurs) voisinage devient réellement problématique car on ne dispose que de très peu de résultats théoriques sur la qualité d'un voisinage pour un problème particulier.

Il existe une mesure de l'adéquation des voisinages appelée rugosité : l'idée consiste à évaluer la variance des coûts de deux solutions voisines. Cette variance est ensuite normalisée par la variance des coûts de l'ensemble des solutions et par la taille du problème, afin d'avoir une mesure indépendante de la valeur absolue de la fonction objectif ou de la taille du problème. [28]

2.6.2 Recherche locale

La recherche locale, appelée aussi la descente ou l'amélioration itérative, représente une classe de méthodes heuristiques très anciennes. Traditionnellement, la recherche locale constitue une arme redoutable pour attaquer des problèmes réputés très difficiles tels que le problème de voyageur de commerce et la partition de graphes. Contrairement à l'approche de construction, la recherche locale manipule des configurations complètes durant la recherche. [29]

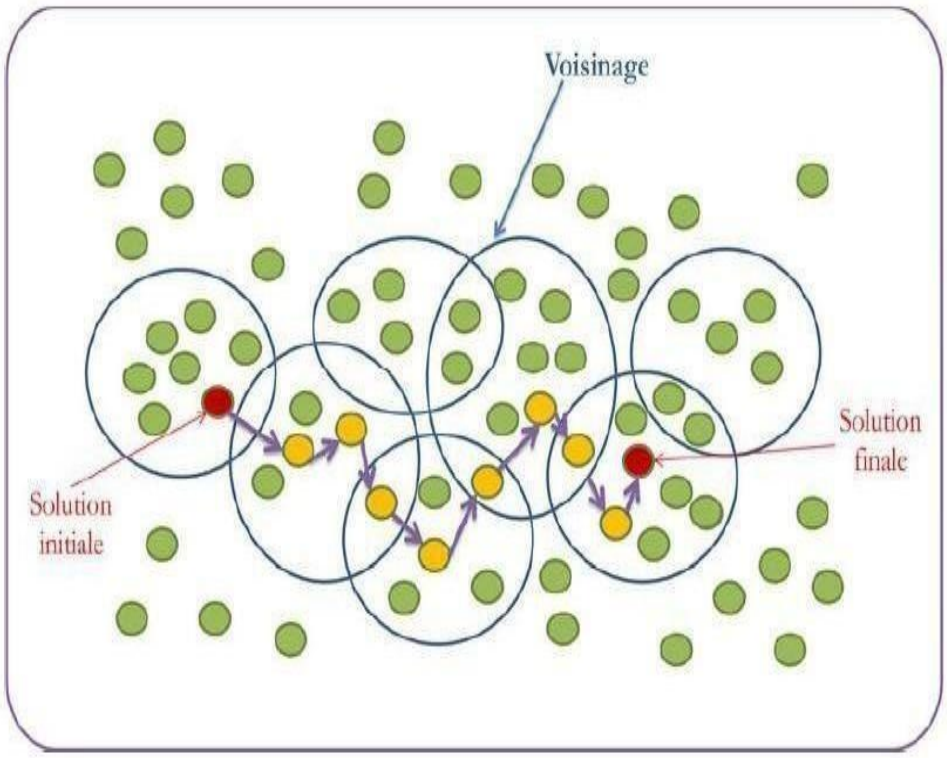


FIGURE 2.2 – Procédure générale de la recherche locale. [2]

2.6.2.1 Principe de recherche locale

Une méthode de recherche locale est un processus itératif fondé sur deux éléments essentiels : Un voisinage :→2 et une procédure exploitant le voisinage. Plus précisément, elle consiste à débuter avec une configuration quelconque s de X, et choisir un voisin s'de s tel que f(s')≺f(s) et remplacer s par s'et à répéter jusqu'à ce que pour tout voisin s'de s, f(s') ≥f(s). Cette procédure fait intervenir à chaque itération le choix d'un voisin qui améliore la configuration courante. Plusieurs possibilités peuvent être envisagées pour effectuer ce choix. Il est possible d'énumérer les voisins jusqu'à ce qu'on en découvre un qui améliore strictement (première amélioration). Comme l'espace des solutions X est fini, cette procédure de descente s'arrête toujours, et la dernière configuration trouvée ne possède pas de voisin strictement meilleur qu'elle-même. Autrement dit, la recherche locale retourne toujours un optimum local. [29]

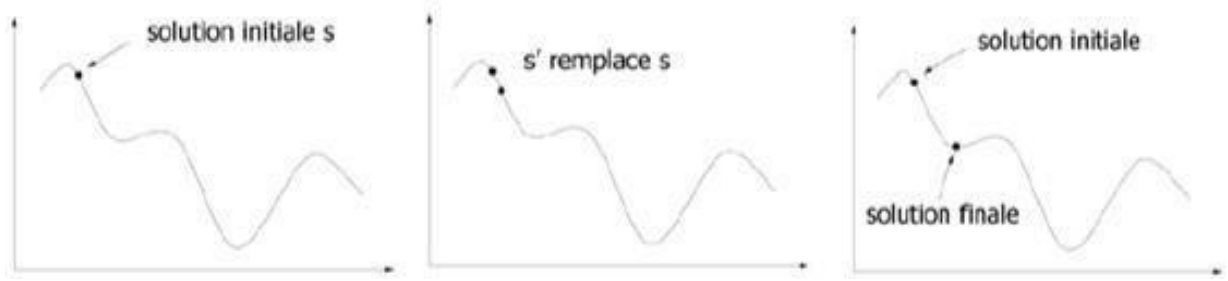


FIGURE 2.3 – Evolution d'une solution dans la méthode de recherche locale. [29]

2.6.2.2 L'algorithme PALS

L'algorithme PALS (de l'anglais ProblemAware Local Search) est une méthode de recherche locale plus rapide et plus compétitif, développé pour résoudre le problème d'assemblage de fragments d'ADN par Alba et Luque (Alba et Luque, 2007). Il représente le pseudocode de cette méthode heuristique. [1]

PALS améliore de façon itérative une seule solution. La solution est codée par une permutation d'entiers représentant les numéros des fragments, ou tous deux fragments successifs se chevauchent. Cette représentation nécessite une liste de fragments assignes chacun à un identificateur (nombre entier) unique. A chaque itération de l'algorithme, la solution courante s'est remplacée par une bonne solution dans son voisinage N (s). La génération de solutions voisines se fait en considérant l'application de mouvements à la solution courante. Un mouvement consiste à inverser l'ordre des fragments situes entre

deux positions distinctes i et j dans la permutations(i.e. Inversion d'une sous-permutation). Ainsi, l'ensemble complet des solutions voisines est défini par la considération de tous les mouvements possibles (pour tout $1 \geq i < j \geq n$, ou n est le nombre de fragments).

Le point clé de PALS est l'utilisation du nombre de contigs comme un critère principal pour juger la qualité d'un résultat d'assemblage de fragments d'ADN. [30]

En revanche, les assembleurs classiques utilisent le niveau de chevauchement entre les fragments successifs comme mesures de qualité, en utilisant des fonctions L'utilisation de telles fonctions pourrait conduire à des situations indésirables dans lesquelles une solution obtenue peut être meilleure qu'une autre solution, avec un grand nombre de contigs (une mauvaise solution). Face à des situations ou plusieurs solutions différentes présentant un même nombre de contigs, les auteurs de PALS ont utilisé le niveau de chevauchement comme un critère secondaire pour évaluer la qualité des solutions. Une solution s est considère meilleure qu'une autre solution s' si et seulement si les conditions suivantes sont vérifiées :

$$nContigs(s) \prec nContigs(s') ou (nContigs(s) = nContigs(s') et F1(s) > F1(s')) [30]$$

Algorithme PALS :

```
1  s ← generateInitialSolution() {créer une solution de départ}
2  répéter
3  |  L ← φ
4  |  pour i ← 1 à n − 1 faire
5  |  |  pour j ← i + 1 à n faire
6  |  |  |  ⟨Δf, Δc⟩ ← calculateDelta(s, i, j) {voir algorithme }
7  |  |  |  si (Δc < 0) ou (Δc = 0 et Δf > 0) alors
8  |  |  |  |  L ← L ∪ ⟨i, j, Δf, Δc⟩ {Mémoriser tous les mouvements non
   |  |  |  |   détériorants}
9  |  |  |  fin
10 |  |  fin
11 |  fin
12 |  si L ≠ φ alors
13 |  |  ⟨i, j, Δf, Δc⟩ ← selectMovement(L) {Sélectionner un mouvement}
14 |  |  applyMovement(s, ⟨i, j, Δf, Δc⟩) {Améliorer la solution}
15 |  fin
16 jusqu'à pas de changement
17 retourner s
```

FIGURE 2.4 – Algorithme : PALS. [21]

Où la fonction nContigs, calcule le nombre de contigs, et la fonction F1, calcule le niveau de chevauchement entre les fragments successifs. [30]

Comme l'opération de calcul du nombre de contigs de chaque solution modifiiez est couteuse en temps de calcul, l'algorithme PALS fait appel à une évaluation incrément ale mesurant le nombre de contigs qui ont et écriés ou supprimés pendant la manipulation des solutions tentatives. Pour chaque mouvement possible, la variation de la longueur totale de chevauchement, notée Δ_f, et la variation du nombre de contigs, notée Δ_c, entre la solution courante et la solution modifiiez après l'application du mouvement sont calculées (voir algorithme CalculateDelta). Le calcul de ces variations exploite les invariants et- nécessite l'analyse des fragments affectes par le mouvement uniquement (i.e., fragments i, j, i 1 et j + 1). [21]

La valeur de Δ_f est calculée par la soustraction de la longueur de chevauchement des fragments affectes de la solution courante de celle de la solution modifiiez (instructions des lignes 3-4 de l'algorithme CalculateDelta). La valeur de Δ_c est calculée en testant, après l'application d'un mouvement, si un contig courant est coupé ou non (les deux premières conditionnelles de l'algorithme CalculateDelta) ou si deux contigs ont été assembles ou non (les deux dernières conditionnelles de l'algorithme CalculateDelta). Ce calcul de Δ_c est base sur la valeur d'un paramètre dit cutoff, qui représente la longueur de chevauchement minimale pour considérer que deux fragments adjacents étant dans le même contig. [21]

A chaque itération de l'algorithme, on calcule les deux critères Δ_f et Δ_c pour tous les mouvements possibles et stocke les mouvements acceptes dans une liste L. On considère uniquement les mouvements qui améliorent la qualité de la solution : ceux qui réduisent le nombre de contigs et ceux qui maintiennent le nombre de contigs et ne diminuent pas le niveau de chevauchement entre les fragments adjacents (la condition de la ligne 7 de l'algorithme PALS). Après avoir stocke les mouvements candidates dans la liste L, on sélectionne un mouvement spécifique et l'applique à la solution courante pour produire une nouvelle solution améliorée. Ce processus est répété jusqu'à aucune amélioration sera possible (i.e., la liste L est vide). [21]

Deux choses restent à déterminer pour compléter la description de l'algorithme PALS : comment générer la solution de départ (la procédure generateInitialSo-lution), et comment sélectionner un mouvement de la liste Là chaque itération (la procédure selectMovement). Les auteurs de l'algorithme PALS, ont conclu, après avoir comparé plusieurs choix possibles,

que la bonne configuration est de généré la valeur de Δ_c. [21]

Fonction : CalculateDelta(s, i, j) :

```
1  Δc ← 0
2  Δf ← 0
   {Calculer la variation du score de chevauchement :}
   {Ajouter le score de chevauchement des fragments affectés de la
    solution modifiée}
3  Δf ← w_{s[i−1]s[j]} + w_{s[i]s[j+1]}
   {Supprimer le score de chevauchement des fragments affectés de la
    solution courante}
4  Δf ← Δf − w_{s[i−1]s[i]} − w_{s[j]s[j+1]}
   {Calculer la variation du nombre de contigs :}
   {Incrémenter le nombre de contigs si un contig est coupé}
5  si w_{s[i−1]s[i]} > cutoff alors
6  |  Δc = Δc + 1
7  fin
8  si w_{s[j]s[j+1]} > cutoff alors
9  |  Δc = Δc + 1
10 fin
   {Decrémenter le nombre de contigs si deux contigs sont fusionnés}
11 si w_{s[i−1]s[j]} > cutoff alors
12 |  Δc = Δc − 1
13 fin
14 si w_{s[i]s[j+1]} > cutoff alors
15 |  Δc = Δc − 1
16 fin
17 retourner (Δf, Δc)
```

FIGURE 2.5 – Algorithme : Fonction CalculateDelta [21]

Dans le cas ou‘ il y a plusieurs mouvements avec la même valeur mini- male de Δ_c, on parfait le choix avec le mouvement ayant la plus grande valeur de Δ_f (ce qui donne une recherche agressive). Plus formellement, un mouvement m donné par i, j, Δ_c, Δ_f est considère meilleur (en termes de degré d'amélioration à réaliser sur la solution courante) qu'un autre mouvement m^J donné par i^J, j^J, Δ_c^J,Δ_f^J si et seulement si les conditions suivantes sont vérifiées :

$$\Delta_c \prec \Delta_c^J ou(\Delta_c = \Delta^J et \Delta_f > \Delta^J)_f \quad [21]$$

2.6.2.3 L'algorithme génétique (AG)

Les algorithmes génétiques (AG) sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Ils fournissent des solutions aux problèmes n'ayant pas de solutions calculables en temps raisonnable de façon analytique ou algorithmique. [30]

Un algorithme génétique est défini par :

- Individu/chromosome/séquence : une solution potentielle du problème.
- Population : un ensemble de chromosomes ou de points de l'espace de recherche.
- Environnement : l'espace de recherche.
- Fonction de fitness : la fonction - positive - que nous cherchons à maximiser ou minimiser.

L'application de l'AG pour résoudre un problème d'optimisation nécessite la définition de :

- Une représentation sous la forme d'un chromosome des solutions (codage). .
- Une fonction objectif pour évaluer la qualité, en termes de fitness, de chaque individu.
- Une méthode d'initialisation de la population des solutions candidates.
- Les valeurs des paramètres de l'algorithme génétique utilisé (par exemple, la taille de la population).
- Les opérateurs génétiques qui produisent l'ensemble des nouveaux individus. s
- Le critère d'arrêt de l'algorithme génétique.

La figure (2.5) décrit le squelette d'un algorithme génétique. Chaque itération de l'algorithme correspond à une génération, où une population constituée de plusieurs individus, représentant des solutions potentielles du problème considéré, est capable de se reproduire. Elle est sujette à des variations génétiques et à la pression de l'environnement qui est simulée à l'aide de la fonction d'adaptation (fitness), ce qui provoque la sélection naturelle (la survie du plus fort).

Les opérateurs de variation sont appliqués (avec une probabilité donnée) aux individus parents sélectionnés, ce qui génère de nouveaux descendants appelés enfants (ou offsprings) ; on parlera de mutation pour les opérateurs unaires, et de croisement pour les opérateurs binaires (ou n-aires). Les individus issus de ces opérations sont alors insérés dans la population. Le processus d'évolution est itéré, de génération en génération, jusqu'à ce qu'une condition d'arrêt soit vérifiée, par exemple, quand un nombre maximum de générations ou un nombre maximum d'évaluations est atteint. [30]

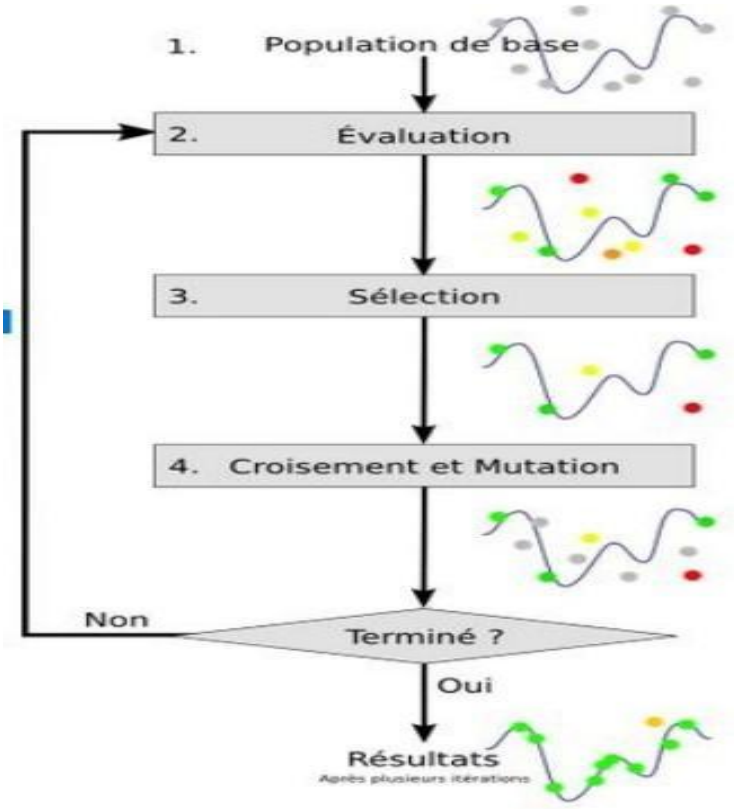


FIGURE 2.6 – Processus d'un algorithme génétique. [31]

2.6.2.3.1 Le codage des données

La représentation des individus est basée sur le codage de l'information, ce dernier concerne les moyens de formaliser l'information afin de pouvoir la manipuler, la stocker ou la transmettre.

Il ne s'intéresse pas au contenu mais seulement à la forme et à la taille des informations à coder.

Pour les algorithmes génétiques, le codage est l'un des facteurs les plus importants, c'est la façon dont elles sont codées les solutions. Le codage est la première étape 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 (un génotype : représente l'ensemble des valeurs des gènes d'un chromosome). Le codage nécessite une adaptation des opérateurs de croisement et mutation.

Il existe deux types de codage :

1. **Le codage binaire :** est la représentation la plus fréquente, ce codage a été le premier à être utilisé dans le domaine des AGs. Il permettre l'utilisation d'opérateurs de croisement et mutation simple.

2. **Le codage réel :** (appelé aussi codage symbolique, ce codage qui j'ai utilisé dans notre recherche) est robuste pour les problèmes considérés comme difficile pour le codage binaire, cela peut être utile notamment dans le cas où l'on recherche le maximum d'une fonction réelle. Si on veut chercher l'optimum d'une fonction à n variables. $f(x_1, x_2 x_n)$, *ou* : x_i Des entiers, alors l'individu I sera de forme :

I :	x_1	x_2	...	x_n

2.6.2.3.2 Génération de la population initiale

L'Algorithme génétique part d'une population de N individus dans le codage choisi (réel), le nombre d'individus influençant significativement la vitesse du programme. La diversité de la population doit être préservée à travers les générations afin d'étudier autant que possible l'espace de recherche. C'est là qu'interviennent les opérateurs de section croisement et de mutation.

2.6.2.3.3 L'évaluation

L'évaluation permet d'assurer que les individus performants seront conservés, alors que les individus peu adaptés seront progressivement éliminés de la population. Elle ne dépendant pas de celle des autres individus.

Il est généralement nécessaire de fournir des valeurs de fitness strictement positives pour permettre aux opérateurs de sélection de fonctionner correctement.

2.6.2.3.4 Sélection

Cet opérateur est important puisqu'il permet aux individus d'une population de survivre, de se reproduire ou de mourir. Il existe plusieurs techniques de sélection.

1. **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.

   P(I) = 1/N (N la taille de la population).

2. **Sélection par tournoi binaire**

   Deux individus sont choisis au hasard et on les fait "combattre", celui qui a la fitness la plus élevée l'emporte avec une probabilité p comprise entre 0.5 et 1.

3. **Sélection par roulette**

   Consiste à donner à chaque individu une probabilité d'être sélectionné proportionnelle à sa performance.

2.6.2.3.5 Croisement

Le croisement utilisé par les algorithmes génétiques est la transposition informatique du mécanisme qui permet, dans la nature, la production de chromosomes qui héritent partiellement des caractéristiques des parents.

L'opérateur de croisement permet la création de nouveaux individus selon un processus fort simple. Il permet donc l'échange d'information entre les chromosomes (individus).

Cet opérateur est appliqué après avoir appliqué l'opérateur de sélection sur la population P, on se retrouve donc avec une population P' de n/2 individus et on doit doubler ce nombre pour que notre nouvelle génération soit complète.

Dans le codage réel l'opérateur de croisement est simple, il suffit de sélectionner un ou plusieurs points de croisement puis fait l'échange de l'information.

1. **Croisement à un point (Codage binaire) :**

   Pour chaque couple, on choisit au hasard un point de croisement. Le croisement s'effectue directement au niveau binaire, et non au niveau des gènes. Un croisement peut être coupé au milieu d'un gène.

2. **Croisement à un deux points (Codage binaire) :**

   On choisit au hasard deux points de croisements successifs. Cet opérateur est généralement considéré comme plus efficace que le précédent.

3. **L'opérateur de croisement (PMX) (Codage réel) :**

   PMX est un opérateur de croisement à deux points de coupure qui définit un segment de même longueur dans chacun des parents P1 et P2.

   Un exemple de l'opérateur de Croisement PMX est illustré à la figure ci-dessous. Les segments de parents (a) ont copié vers les enfants E1 et E2. E1 a hérité du segment de P2 et E2 de P1(b).

4. **L'opérateur de croisement (CX) (Codage réel)**

   CX est un opérateur qui satisfait la condition suivante : chaque gène d'un enfant provient de l'un des parents à la même position. Les enfants sont donc formés en copiant un gène d'un parent et en éliminant l'autre à la même position puisqu'il va appartenir au deuxième enfant. Une fois que les positions occupées sont copiées par élimination, on a complété un cycle. Les places restantes des deux enfants sont complétées par les parents opposés. (Voir l'exemple suivant)

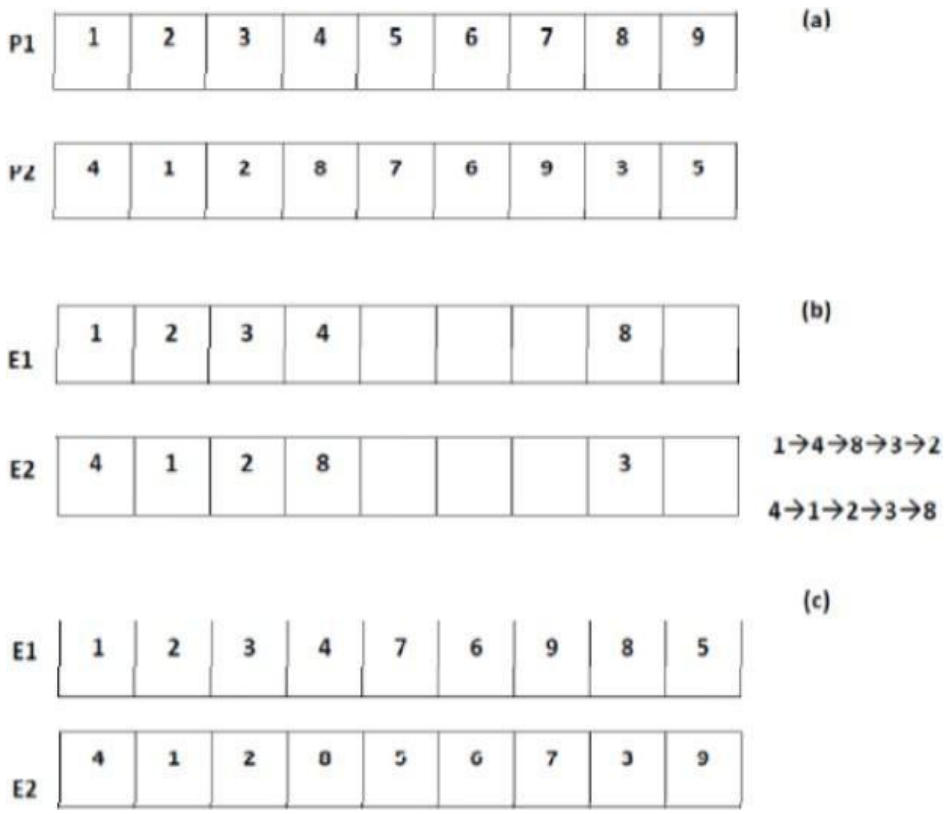


FIGURE 2.7 – Représentation schématique du croisement CX. [8]

2.6.2.3.6 Mutation

La mutation est un changement aléatoire selon une certaine règle probabiliste qui doit faire sur les génotypes, avec une faible probabilité Pm (fixée par l'utilisateur), La probabilité P_m de mutation selon Fonseca est calculée comme suivant :

$$P_m = 1 - \sigma^{-1/l} \tag{2.3}$$

l : est la longueur du chromosome.

σ : est la pression sélective, où sa valeur recommandée est où sa valeur recommandée est : 1,8.

La mutation classique consiste à transformer dans un chromosome binaire un 1en un 0 ou le contraire.

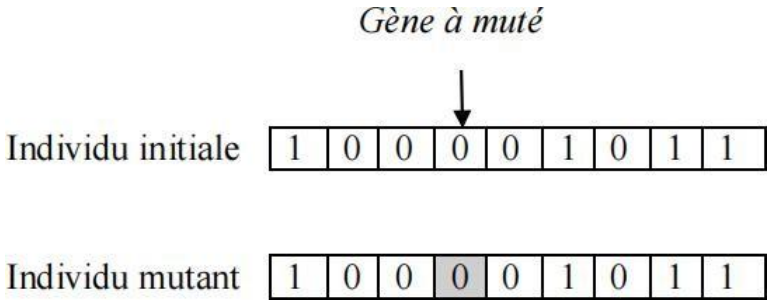


FIGURE 2.8 – Représentation schématique de mutation. [8]

Pour le codage réel, les opérateurs de mutation les plus utilisés sont les suivants :

1. **Mutation par inversion :**

   Deux positions sont sélectionnées au hasard et tous les gènes situés entre ces positions sont inversés.

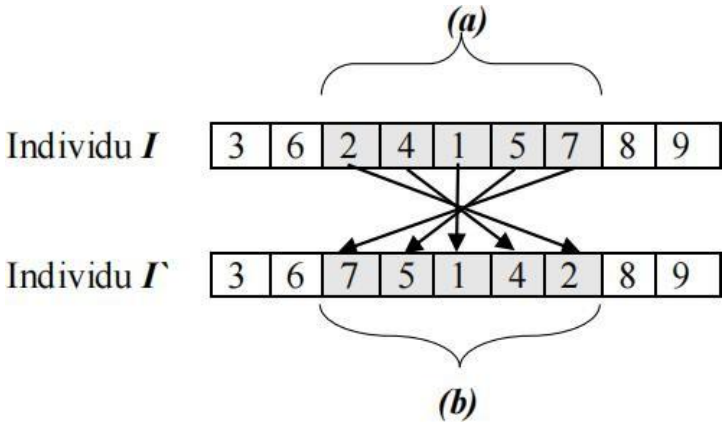


FIGURE 2.9 – Représentation schématique de la mutation par inversion.

2. **Mutation par déplacement :**

   Une séquence est sélectionnée au hasard et déplacée vers une position elle-même tirée au hasard.

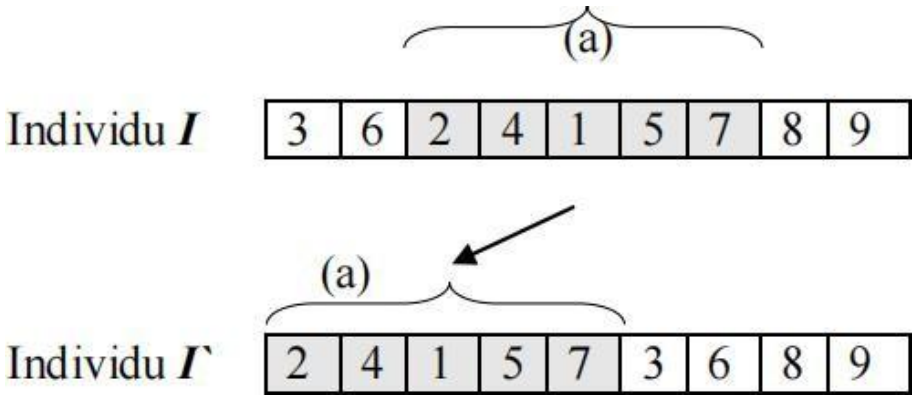


FIGURE 2.10 – Représentation schématique de la mutation par déplacement. [8]

3. **Mutation par permutation :**

   Deux positions **P1** et **P2** sont sélectionnées au hasard et les gènes situés dans ces positions sont permutés.

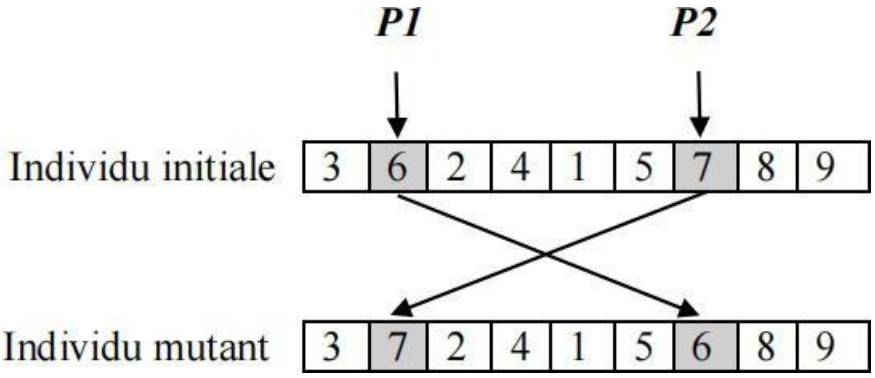


FIGURE 2.11 – Représentation schématique de la mutation par permutation. [8]

2.6.2.3.7 Remplacement

Cette étape sert à remplacer la population de l'instant t (itération t) par la nouvelle population de l'instant t+1. C'est la dernière étape de chaque itération jusqu'au critère d'arrêt suffisant.

Une stratégie élitiste consiste à conserver au moins un individu de la population de la génération précédente, dans la génération suivante.

2.6.2.4 PALS avec l'algorithme Génétique

Alba et Luque ont proposé dans d'appliquer l'algorithme PALS comme operateur de mutation dans un GA. Dans ce schéma hybride, la population de GA est exploitée pour fournir à l'algorithme PALS de multiples configurations de départ. [21]

Alba et Dorronsoro, 2009b, Dorronsoro et al., 2008). Pour le meme but, Minetti et al. ont recemment proposé dans (Minetti et al., 2014) une méthode distribuée combinant l'algorithme PALS avec un algorithme de recuit simule. L'objectif était de pouvoir traiter les données bruitées. Dans cette approche hybride, l'algorithme de recuit simulé est utilisé pour fournir des configurations de départ pour l'algorithme PALS, tandis que le modèle distribue est adopté afin de promouvoir la diversification de recherche. [21]

2.7 Conclusion

Dans ce chapitre, nous avons présenté le principe de l'algorithme de recherche local PALS avec algorithme génétique pour résoudre notre problème. Le chapitre suivant sera consacré à la conception et l'implémentation de cet algorithme et la description des résultats obtenus.

Chapitre 3

Conception et implémentation

3.1 Introduction

Notre objectif est de développer un algorithme hybride pour l résoudre un problème d'assemblage de fragments d'ADN NP-difficile. Cette hybridation permet de combiner un algorithme à évolution différentielle d'optimisation avec un algorithme de recherche locale PALS.

Nous avons subdivisé notre chapitre en cinq sections, nous allons commencer par description du problème de travail, ensuite nous allons expliquer le cycle de projet, ensuite la description détaillée de PALS et l'algorithme génétique et la dernière section dédiée à la conception d'algorithme hybride.

3.2 Description global et détails du système

Dans ce travail nous avons utilisé l'algorithme génétique pour résoudre ces deux problèmes : utiliser AG pour sélectionner le meilleur mouvement dans l'algorithme PALS et appliquer la solution obtenue par AG pour réinitialiser PALS.

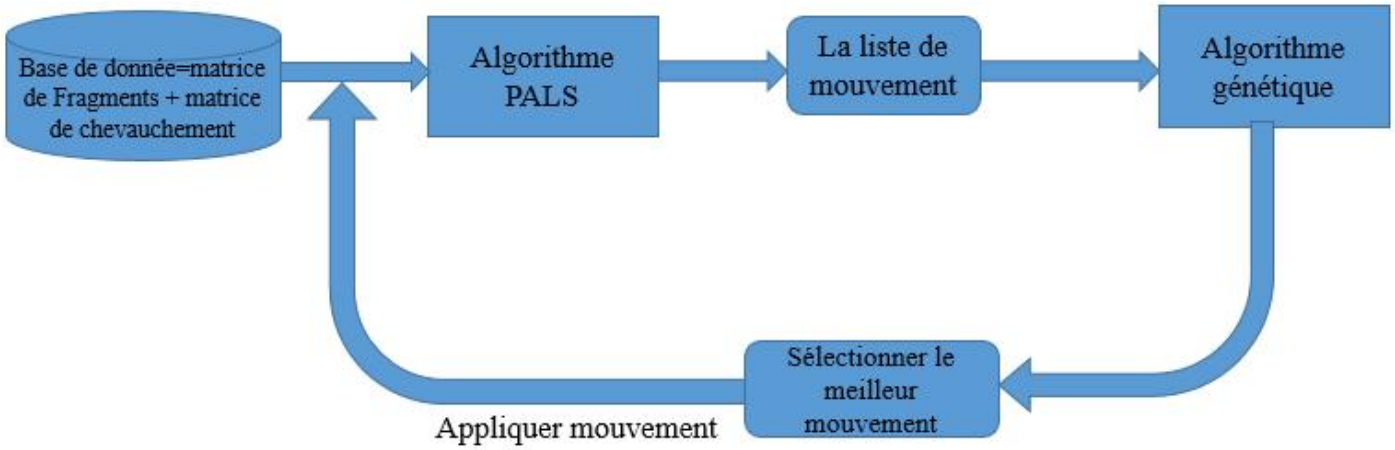


FIGURE 3.1 – Description global du système.

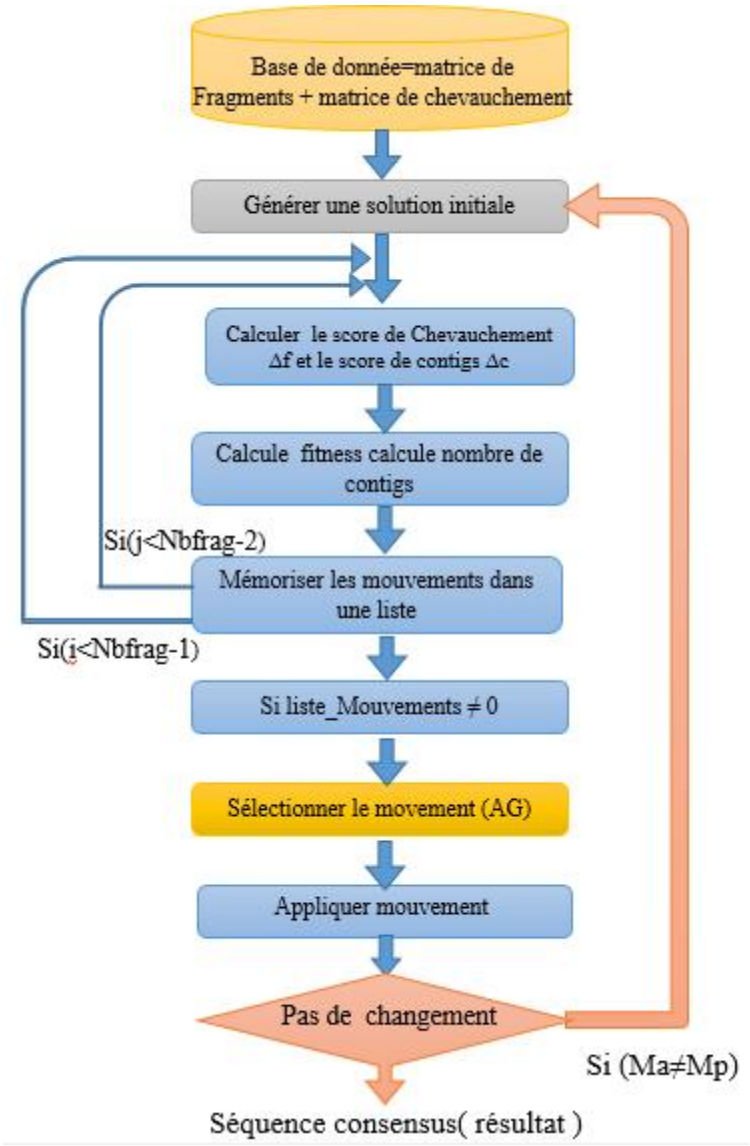


FIGURE 3.2 – Description détails du système

- Ma: Mouvement actuel
- Mp: Mouvement précédent

3.3 Environnement de développement

Pour l'implémentation nous avons choisi l'environnement Matlab de notre assembleur de fragment d'ADN :

3.3.1 MATLAB

Est un langage de script émulé par un environnement de développement du même nom ; a été conçu par Cleve Moler à la fin des années 1970 ; il est utilisé à des fins de calcul numérique. Développé par la société The MathWorks, MATLAB permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en uvre des algorithmes, de créer des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, C++, Java,éet Fortran.

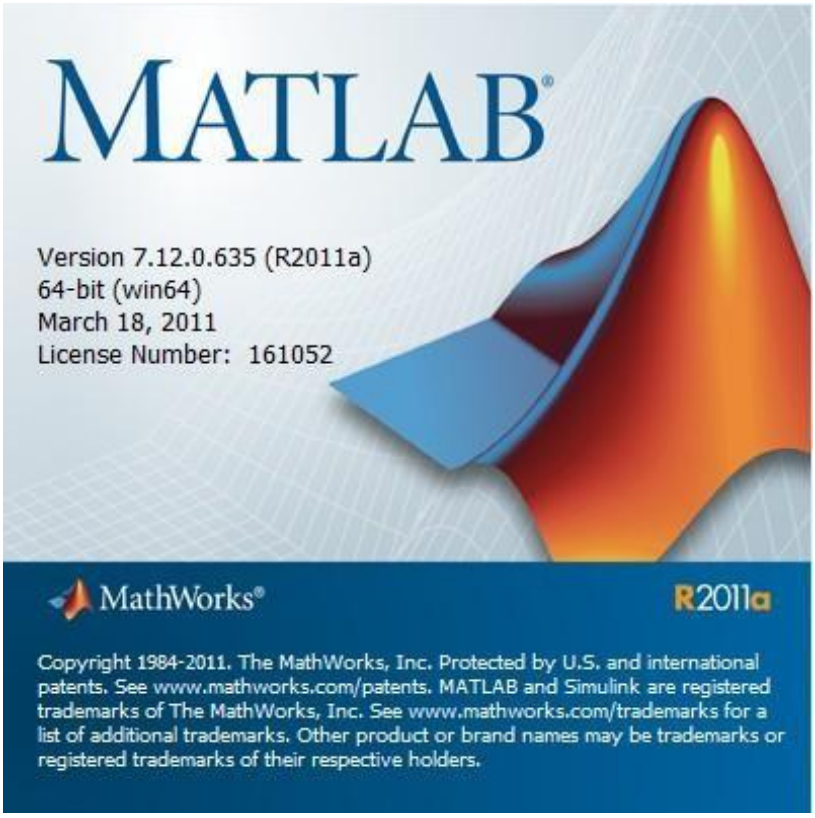


FIGURE 3.3 – MATLAB.

Les algorithmes étudies sont implémentes en utilisant le paradigme de logiciels et de

matériel qui sont résumés dans le tableau (3.1) :

Matériel	Version
OS	Microsoft Windows 10 Professional, 64bits
CPU	Intel(R) Celeron(R) CPU N3060 @ 1.60GHz 1.60 GHz
RAM	4.00Go

TABLE 3.1 – Les caractéristiques de Matériel

3.3.2 Base de données utilisée

Nous avons utilisé les instance d'ADN fournis par l'outil GenFrag , il est conçu comme mesure de comparaison des assembleurs (algorithme d'assemblage) en termes d'efficacité et de qualité de solution, il génère artificiellement des instances d'ADN. Ces benchmarks sont issus d'un travail de L.Michael Engle et Christian Burk [2]. Les fragments générés possèdent un seuil minimal de chevauchement égal à 30pb, ce qui rend l'assemblage un peu difficile pour les instances de grande taille. Les instances de GenFrag.

Le tableau (3.1) présente des informations sur les ensembles de fragments spécifiques que nous utilisons pour tester notre algorithme.

Instance	Moyenne de longueur des fragments	Nombre de fragments	Couverture	La langueur de la séquence
X60189-4	395	39	4	
X60189-5	286	48	5	
X60189-6	343	66	6	3835
X60189-7	387	68	7	
M15421-5	398	127	5	
M15421-6	350	173	6	10089
M15421-7	383	177	7	
j02459-7	405	352	7	
38525243_4	708	442	4	77292
38525243_7	703	773	7	

TABLE 3.2 – Base de données utilisée GenFrag.

GenFrag prend une séquence d'ADN laosn et l'utilise comme brin parent à partir duquel des fragments aléatoires sont générés selon les critères fournis par l'utilisateur (longueur moyenne des fragments et couverture de la séquence parent). Nous avons choisi une grande séquence du site web du NCBI : une région ADN du CMH humain de classe II avec des

répétitions de fibronectine de type II HUMMHCFIB, avec le numéro d'accession X60189 ; un apoloprotéme humain HUMAPOBF, avec le numéro d'accession M15421 ; le génome complet du bactériophage lambda, avec le numéro d'accession J02459 ; et une séquence du BAC de Neurospora crassa, avec le numéro d'accession .38524243'.

Nous avons utilisé deux fichier (fichier.dat,fichier.csv) :

- **Fichier.dat :** présent la liste de fragments.
  Les fragments utilisés sont représentés dans un tableau multidimensionnel contient deux champs (le numéro de fragment, contient le fragment lui-même).
  La commande Matlab l'accès fréquent au fichier de fragments est :

```
\% read fragments file
[filename, pathname,] = uigetfile('*.dat', 'Choisir le fichier
de fragments');
name = [pathname filename];
[Header,Sequence] = fastaread(name)
```

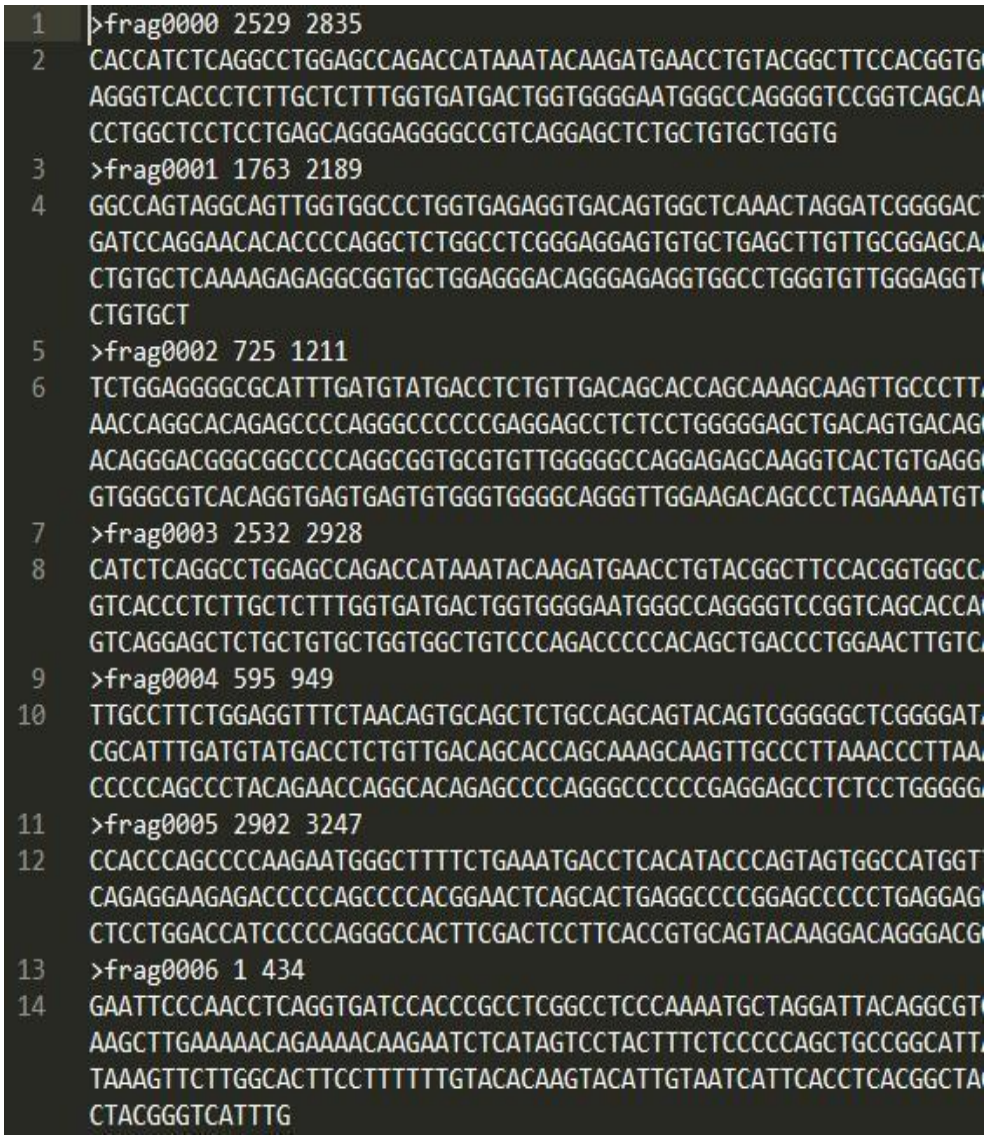


FIGURE 3.4 – Liste de fragments.

- **Fichier.csv :** présent la liste de chevauchement entre les fragments.
  Les chevauchements entre les fragments sont représentés par Une matrice carrée de taille n x n Figure (3.3).

  n : est le nombre de fragments.

  La commande Matlab l'accès fréquent au fichier est :

```
[filename, pathname,] = uigetfile('*.csv', 'Choisir le fichier chevach');
name = [pathname filename];
ftoread = name
fid = fopen(name, 'r')
```

	A	B	C	D	E	F	G	H	I
1	# Generated by ../matriz.pl date: Tue Sep 17 10:38:03 2013								
2	# pairs file: pairs_conservative.txt symetric matrix 39 fragments								
3	0	11	48	304	10	12	13	11	10
4	11	0	23	11	20	11	9	331	302
5	48	23	0	48	225	193	8	23	23
6	304	11	48	0	13	27	13	11	10
7	10	20	225	13	0	107	13	11	10
8	12	11	193	27	107	0	15	11	11
9	13	9	8	13	13	15	0	9	30
10	11	331	23	11	11	11	9	0	302
11	10	302	23	10	10	11	30	302	0
12	163	9	255	160	48	124	8	9	26
13	247	11	10	327	10	14	13	11	10
14	48	23	178	48	12	10	33	23	23
15	20	11	262	43	107	346	15	11	11
16	9	18	14	9	17	8	9	18	18
17	13	9	8	13	13	15	414	9	30
18	19	9	49	19	8	8	9	9	11
19	51	9	210	51	10	88	9	9	9
20	8	13	149	13	279	26	13	13	10
21	51	11	322	51	107	241	8	11	11
22	14	281	13	14	20	11	9	185	156
23	13	9	8	13	13	15	301	9	9

FIGURE 3.5 – Chevauchement entre les fragments.

3.3.3 Généré une solution initiale

L'objectif de cette fonction est généré aléatoirement une solution initiale. L'ensemble de fragments numérotés de 1 à NbFrag, Exemple : S comport 39 fragment : Ensemble de 39 fragments. Alors (NbFrag=39) numérotés de 0 jusqu'à 38. S={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14, 15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31, 32,33,34,35,36,37,38}

La commande Matlab randperm de générer aléatoirement un fragment de tai le NbFrag (NbFrag : le nombre de fragments) est la suivante :

```
solution = randperm(NbFrag) ;
```

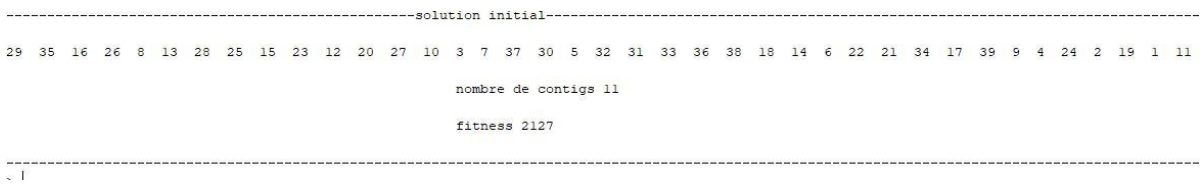


FIGURE 3.6 – La solution initiale.

3.3.4 Calculer le score de chevauchement Δ_f et le score de contigs Δ_c :

L'objectif de la fonction Calcul_deltaF_deltaC permet d'est permet calculer le score de chevauchement Δ_f et le score de contigs Δ_c, il est possible de trouver une séquence avec le plus grande de score de chevauchements mais avec un nombre élevé de contigs.

Δ_c : Si un contig courant est coupé ou non après l'application d'un mouvement en se basant sur la valeur d'un paramètre cutoff.

Δ_f : Si une valeur de score calculé par la soustraction de la longueur de chevauchement des fragments affectés de la solution courante.

PALS utiliser les deux métrique Δ_c, Δ_f pour évaluer la qualité des modifications.

3.3.4.1 Calculer de Fitness

Dans cet algorithme Chaque individu de solution est évalué par la fonction fitness l'objectif de cette fonction est qui représente la performance de solution.

La fonction d'objectif (fitness) calcule la somme de chevauchement entre deux fragments de la solution S, de deux fragments successifs dans la solution.

$$F(s) = \sum_{i=1}^{n-1} w_{s[i],s[i+1]}$$

s : une solution (individu) à l'instant t

n : nombre de fragments

$s_{[i]}$,$s_{[i+1]}$: deux fragments successif

$w_{s[i]}, s_{[i+1]}$: chevauchement entre les fragments $s_{[i]}$ et $s_{[i+1]}$

Fonction 3.1 : Calcul_fitness (S,NbFrag, chevauch)

S : solution

NbFrag : nombre de fragments

Chevauch : matrice de chevauchements

Début

fitness ←0 ; { fitness de la solution}

Pour i=1, NbFrag 1 faire

fitness ←fitness + chevauch(S(i),S(i+1))

Fin pour

Retourne fitness {fitness de la solution}

Fin

Exemple illustratif :

Supposant, le nombre de fragments égale à 9 :

Avec les fragments suivants :

F0 : GTCAATT	***F1*** : CTAGGAACTA	***F2*** : TGCG
F3 : TTACATTGCG	***F4*** : CCAATTGA	***F5*** : AACTATGT
F6 : CGTAATTGCC	***F7*** : GATACCG	***F8*** : CGTAATT

On calcule la fitness pour chaque solution.

Solution :

1	5	0	3	2	6	4	7	8

Le schéma suivant, présenté l'alignement des 9 fragments de la solution 1 :

F1 : CTAGGAACTA
F5 : AACTATGT
F0 : GTCAATT
F3 : TTACATTGCG
F2 : TGCG
F6 : CGTAATTGCC
F4 : CCAATTGA
F7 : GATACCG
F8 : CGTAATT

$f(S1)$= chevauchement(1,5) + chevauchement(5,0) +
chevauchement (0,3) + chevauchement(3,2) +
chevauchement(2,6) + chevauchement(6,4) +
chevauchement(4,7) + chevauchement(7,8)

=1+2+2+4+2+2+2+3=**18**

3.3.4.2 Calculer le nombre de contigs

L'objectif de cette fonction est permet de calculer le nombre de contigs.

Fonction 3.2 : Calcul_nombre_contigs (S, chevauch, cutoff)

S : solution

Chevauch : matrice de chevauchements

cutoff : seuil de chevauchement

Début

Nombre_contigs ←1 nombre de contigs de la solution S

Pour i ←1,NbFrag-1 faire

si chevauch(S(i),S(i+1)) ⇐cutoff alors

Nombre_contigs ←Nombre_contigs + 1

Fin si

Fin Pour

Retourne nombre_contigs

Fin

Exemple 1 :

Solution : cutoff = 2

3	0	2	1

F3: CGTAATT
F0: AACTATGT
F2: GTCAATT
F1: TTACATTGCG

- Le nombre de contigs est 2.
- Le nombre de fitness est 4.

Exemple 2 :

Solution : cutoff = 2

0	2	3	1

- Le nombre de contigs est 1.
- Le nombre de fitness est 6.

3.3.5 Mémoriser les mouvements dans une liste

L'objectif de cette fonction est de mémoriser les meilleurs mouvements dans une liste.

Fonction 3.3 : Liste_mouvement (i,j,Delta_F, Delta_C)

S : solution

Début

Liste_mouvement(numMouv)←solution

Liste_mouvement(i)←i

Liste_mouvement(j) ←j

Liste_mouvement (Delta_F)←Delta_F

Liste_mouvement(Delta_C)←Delta_C

Fin

3.4 Sélectionner un mouvement par (AG)

3.4.1 Génération de la population initiale

Si la List L de mouvement (la sortie de PALS) n'est pas vide. Nous choisir une population initiale de taille 6.qui représente les 6 premiers mouvements après la classification.

3.4.2 Evaluation de la population

Chaque individu de la population est évalué par le calcul fonction objectif (fitness) et nombre de contiqe.

3.4.3 Sélection

Cet opérateur est peut-être le plus important puisqu'il permet aux individus d'une population de survivre, de se reproduire ou de mourir. Dans notre projet, nous avons choisi la sélection par tournoi, son principe est le suivant :

Pour sélectionner un individu, on en tire t (t la taille du tournoi) uniformément dans la population, et on sélectionne le meilleur de ces t individus.

```
112 -       for tr=1:Nombre_Iteration
113 -       for i=1:6
114 -       pop(j)= popilation(i);
115
116 -       if  j==6
117 -           j=1;
118 -       end
119
120 -       if  j == 3
121
122 -      parent(1)= pop(1);
123 -      parent(2)= pop(2);
124 -      parent(3)= pop(3);
125
```

FIGURE 3.7 – Sélection d'une population initiale

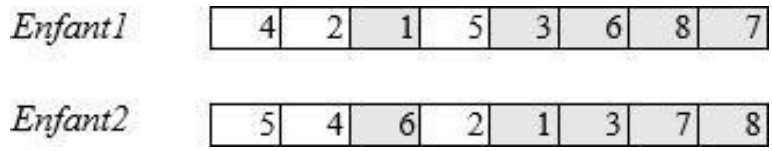
3.4.4 Croisement

Dans notre projet, nous avons appliquée l'opérateur de croisement CX (Cycle Crossover), pour le codage réel, dont le principe est illustré dans l'exemple suivant :

On a deux solutions parentes (solutions) **parent1** et **parent2** :

Parent1	4	2	6	5	1	3	7	8
Parent2	5	4	1	2	3	6	8	7

Après l'application d'opérateur CX :



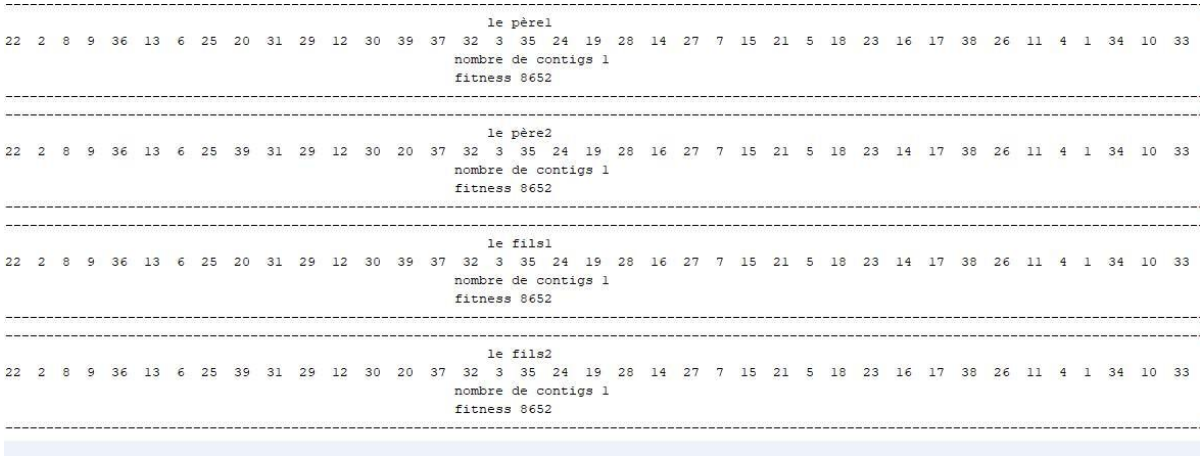


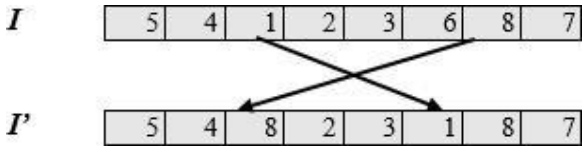
FIGURE 3.8 – Croisement (CX)

3.4.5 Mutation

Nous avons appliqué l'opérateur de mutation par permutation à chaque enfant. Le principe de la mutation par permutation est :

- Générer deux nombres aléatoires pt1 et pt2 ; pt1, pt2 (1, n), n est le nombre de fragments.
- Faire une permutation entre les gènes pt1 et pt2.

On a l'individu I, et deux points (pt1, pt2) générés de façon aléatoire, par



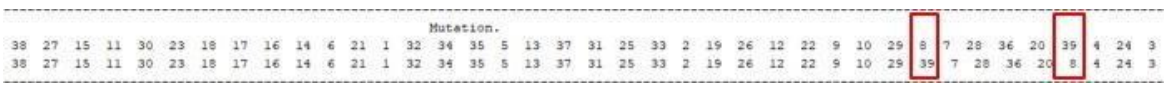


FIGURE 3.9 – Mutation.

3.4.6 Sélectionner le meilleur mouvement

```
910
911 -          for x=1:length(SelectMovment)
912 -              MeilleureMovment=SelectMovment(1);
913 -              if   SelectMovment(x).nContigs < MeilleureMovment.nContigs
914 -                  MeilleureMovment=SelectMovment(x);
915 -              else
916 -                  if   SelectMovment(x).nContigs == MeilleureMovment.nContigs
917 -                   if   SelectMovment(x).fitnes > MeilleureMovment.fitnes
918 -                      MeilleureMovment=SelectMovment(x);
919
920 -                  end
921 -                  end
922 -               end
923
924
925 -          end
926 -          end
```

FIGURE 3.10 – Le meilleur mouvement

3.4.7 Critère d'arrêt

Le critère d'arrêt est considéré jesqu pas de changement c'est-à-dire pas de changent entre la solution courante et la solution modifiée.

3.5 Expérimentation et Résultats

Exemple :

Les figures (3.10)présentent le résultat d'assemblage de fragments de l'instance X60189-4 PALS avec AG par :

- Cuttoff est 10.
- Nombre de population est 10.
- Nombre d'itération est 10.

La solution finale :

```
--------------------------------------------------Meilleur Mouvment--------------------------------------------------------------------------------------------------
29  12  1  4  8  9  36  7  15  21  27  23  18  5  38  32  3  35  24  34  10  33  19  17  16  14  28  2  22  31  37  26  13  6  25  20  30  39  11

                                                nombre de contigs 1

                                                fitness 9225

----------------------------------------------------------------------------------------------------------------------------------------------------------------
```

FIGURE 3.11 – La solution finale.

Le tableau (3.3) présente les résultats obtenus après l'application PALS vs PALS avec AG.

Le valeurs cutoff est 30.

L'instance		PALS			PALS avec AG			
Nom d'instance	Nombre de fragments	Contig	Fitness	Temps d'exécution (s)	Nombre d'itération	Contig	Fitness	Temps d'exécution (s)
X60189-4	39	3	8865	2.440445 E +01	10	7	6092	2.862097 E +01
					15	2	3279	4.505818 E +01
X60189-5	48	7	12053	6.544889 E +01	10	6	2600	5.7902255 E +01
					15	5	4257	4.46243 E +01
X60189-6	66	9	13218	1.940873 E +02	10	9	3223	9.684079 E +02
					7	7	8894	2.138604 E +02
X60189-7	68	3	16584	1.826313 E +02	10	6	13194	7.765832 E +02
					13	2	10432	7.765832 E +02
m15421-5	127	37	29552	5.220721 E +03	12	30	8283	7.765832 E +01
					8	42	10879	3.379320 E +03

TABLE 3.3 – Comparaison entre PALS vs PALS avec algorithme génétique.

Nous concluons du tableau (3.3) que la performance du complexe PALS avec AG sur le seuil d'interférence (cutoff), dépend de la taille de la communauté et du nombre d'itérations.
L'hybridation d'un algorithme PALS avec un algorithme génétique (AG) améliore les résultats de clustering (dans quelques cas), mais augmente le temps d'exécution.

3.6 Conclusion

Dans ce chapitre, nous avons présenté le thème de notre projet. Nous avons décrit notre résolution au problème d'assemblage de fragments d'ADN à l'aide de l'hybridation entre PALS et AG avec des exemples illustratifs.
Ainsi que l'implémentation de ces algorithmes et la description des résultats obtenus.

Conclusion général

Dans notre projet nous avons traité le problème d'assemblage des fragments d'ADN avec l'algorithme génétique AG et l'algorithme PALS, nous avons conclu que la performance d'un assembleur d'ADN conçu par un algorithme pals dépend de la solution initiale et nombre d'itérations (si on augmente le nombre d'itérations, on obtient un score d'alignement grand).

Alors que la performance de l'assembleur PALS avec AG dépend du seuil de chevauchement (cutoff), dépend de la taille de la population et le nombre d'itérations. L'hybridation de l'algorithme PALS avec l'algorithme génétique (AG) améliore les résultats d'assemblage (dans quelques cas), mais augmente le temps d'exécution. Mais pour améliorer le temps de traitement des grandes instances et assurer très rapidement sa convergence, nous envisagerons une ré-conception de notre système pour l'implémenter et l'exécuter sur des machines parallèles, ceci reste comme une perspective.

Bibliographie

[1] Abdelkamel Ben Ali, Gabriel Luque, Enrique Alba, Kamal Melkemi, "An improved problem aware local search algorithm for the DNA fragment assembly problem". Article (journal Soft Comput). 1 April 2017

[2] Rauof Bouhali, Yacin Djerroud,Sider Abderhamen. "Algorithme de recherche locale pour le problème d'assemblage de fragments d'ADN sur processeurs graphiques (GPU)".Mémoire de master.Université Abderrahmane Mira-Bejaia. 2013.

[3] https://www.elysia-bioscience.com/voyage-a-travers-ladn-nucleaire-sans-oublier-ladn-mitochondrial/ "Voyage à travers l'ADN nucléaire sans oublier l'ADN mitochondrial.png"

[4] Nathalie Nicole Poirier. "L'utilisation de la preuve par l'ADN et ses impacts sur notre société". Université de sherbrooke. 2014

[5] Alain Boudet, "L'ADN électromagnétique et la communication entre cellules". 1 septembre version augmentée .Article (journal copyright).30 avril 2014

[6] Camus Camus, "Les rôles de l'ATP". Document de cours de l'université de Montréal. 2011

[7] Sarah Bourgoin, "Protocoles rapides pour la préparation d'ADN à partir d'échantillons Caractéristique de la biologie judiciaire". Mémoire, Université du Québec à Trois- Rivières, décembre 2000.

[8] Boudouh Nouara. "Optimisation par algorithme génétique pour la résolution du problème d'assemblage de fragments d'ADN". Mémoire de master. Université Mohamed Khider BISKRA 2019.

[9] Enrique Alba , Gabriel Luque, "A new local search algorithm for the DNA fragment assembly problem". Article (Journal Evolutionary Computation in Combinatorial Optimization). April 11 13 2007.

[10] Michael Engle, Christian Burks, GenFrag. "New features for more robust fragment assembly benchmarks". Thèse Oxford University Press. September 1994.

[11] https://th.bing.com/th/id/OIP.6j8OhoZB3gr5jhcpyggORQHaEK ?pid=ImgDetrs=1.

[12] Levene.MJ, Korlach.J,Turner.SW, Foquet.M et HG. Craighead.H, Zero-mode "Waveguides for single-molecule analysis at high concentrations". Article(Journal science). Jan 2003

[13] Pierre Alain Brailiard, "Enjeux philosophiques de la biologie des systèmes". thèse de Doctorat. Université Paris 1 Panthéon Sorbonne. Octobre 2008.

[14] Michel Delarue et Gilles Furelaud. "Le séquençage d'un ADN". Article. Samedi 1 mai 2004, Dernière modification Mardi 20 mars 2018

[15] https://www.oezratty.net/wordpress/wp-content/WindowsLiveWriter/Les-dessous-techniques-du-squenage-du-gn_82A7/ABI-Solid_thumb.jpg.

[16] Charrat.N, "Diagnostique moléculaires et application en agroalimentaire caractérisation D des souches de campylobacter isolées à partir du poulet de chair au Maroc". Thèse universitaire Université Mohammed.V, Rabat. 19 mai 2017.

[17] Sengenès.J, "Développement de méthodes de séquençage de seconde génération pour l'analyse des profils de méthylation de l'ADN". Université Paris. 30-03-2012.

[18] Corem.S, "Le séquençage à haut débit". Livre. Université Pierre et Marie Curie, 062012.

[19] Chadi Saad, "Caractérisation des erreurs de séquençage non aléatoires". Thèse de Doctorat. Université de Lile, Septembre 2018.

[20] https://almerja.net/medea/images/2x_146.jpg.

[21] Ben Ali.A, "Contributions à la résolution de problème d'optimisation combinatoire NP-difficile". Thèse de Doctorat .Université Mohamed Khider Biskra, 19 Avril 2018

[22] Lucien MOUSIN. "Extraire et exploiter la connaissance pour mieux optimiser". Thèse de Doctorat. mardi 22 janvier 2019

[23] Cheniguel Lokmane, Drouiche Amina Ghanem.S. "Algorithmes Bio-inspirés pour l'optimisation du Routage dans les Réseaux Ad Hoc (Doctoral dissertation". Univ. A/Mira Bejaia. 2020.

[24] Ghrib Ramadhan.A "Hybrid Metaheuristic Algorithm for Solving the DNA Fragment Assembly Problem". Mémoire de Master. Université d'El-Oued. 2014

[25] https://waytolearnx.com/2019/03/difference-entre-un-probleme-np-complet-et- np difficile.html 14/05/2022

[26] Hassani nour elhouda. "Traitement parallèle pour l'assemblage de fragments d'ADN". Mémoire de master. Université Mohamed Khider BISKRA .2021

[27] https://fr.wikipedia.org

[28] Slimani.L. "Contribution à l'application de l'optimisation par des méthodes métaheuristiques à l'écoulement de puissance optimal dans un environnement de l'électricité dérégulé". Thèse de Doctorat. Université de Batna 2. 2009.

[29] Kherbouche.L, Oubahri.Z. "Quelques méthodes de résolutions en optimisation combinatoire". Mémoire de master. Université Mouloud Mammeri, TIZI OUZOU.2017

[30] Djerou.L, "Algorithmes génétiques", Cours du module AAO -Algorithmes Avancés et optimisation-, Université Mohamed Khider BISKRA, 2022.

[31] https://upload.wikimedia.org/wikipedia/commons/4/42/Schema_simple_algorithme_genetique.png.