Jérôme AUDOUX
Promotion 2014

IRB - INSERM
CHU Saint-Eloi

RAPPORT DE STAGE

MASTER 2 STIC POUR LA SANTÉ

SPÉCIALITÉ **B**IOINFORMATIQUE **C**ONNAISSANCES ET **D**ONNÉES

15 juillet 2014

La reconstruction du transcriptome à partir des données de RNA-Sequencing

---

Tuteur pédagogique

Corinne LAUTIER
MAÎTRE DE CONFÉRENCES À
L'UNIVERSITÉ MONTPELLIER 2

Tuteur laboratoire

Nicolas PHILIPPE
POSTDOCTORAL RESEARCHER IN
BIOINFORMATICS

Table des matières

Remerciements

Je tiens, tout d'abord, à remercier Nicolas Philippe, pour m'avoir épaulé et accompagné dans ce travail de recherche. Sa grande générosité, son humanité et ses précieux conseils, au cours de la phase de préparation et de rédaction de ce mémoire, ont été réellement bénéfiques. Je remercie également Thérèse Commes pour sa spontanéité, son attention et son engagement à mon égard.

Il me paraît maintenant indispensable de saluer Isabelle Chuine, qui a éveillé ma curiosité et mon intérêt pour la bio-informatique, au cours d'un premier travail de recherche, il y a maintenant trois ans.

Je tiens en outre à remercier Alban Mancheron pour son amitié et pour m'avoir ouvert les portes de son Master.

Je tiens également à remercier tous mes collègues - et tout particulièrement Florence Ruffle et Anthony Boureux, avec qui j'ai eu la joie de partager cette année - pour leurs encouragements, leur gentillesse, et leur présence sans faille.

Il est désormais temps de remercier chaque membre de laboratoire de l'IRB qui ont fait que chaque jour se déroule dans la joie et la bonne humeur, tout au long de l'année.

Merci également aux logiciels libres qui ont permis à ce rapport d'exister : Latex, Pandoc, Markdown markup langage, Git et Vim.

Je souhaites remèrcier mon commité oficiel de corrections composé de Jésus, Alizée, Nicolas et Thérèse sans qui mes phrase ressembleraits à celle çi.

Enfin, je souhaite témoigner tout particulièrement ma reconnaissance à Alizée, ma compagne de toujours, qui m'a soutenue et accompagnée sans réserve, tout au long de la rédaction de ce mémoire et qui rédige actuellement mes remerciements.

Introduction

L'ADN contenu dans chacune de nos celulles contient l'ensemble de l'information génétique qui a permis à chacun d'entre nous de nous développer d'un être unicellulaire jusqu'à une forme de vie très complexe composée de quelques cent mille milliards de cellules, chacune différenciée en plusieurs centaines de types cellulaires. L'ensemble de cette diversité est liée à des mécanismes biologiques qui régissent l'expression de l'information génétique et permettent le développement et le fonctionnement des cellules. Cependant, etudier le *génome* n'est pas suffisant pour comprendre toute la diversité biologique, c'est pourquoi les biologistes portent une grande attention au *transcriptome* qui est une couche de complexité se plaçant au dessus du génome. En effet, certaines zones du génome vont être transcrites afin de jouer un rôle "actif" dans la cellule, les transcrits, et ce sont ces derniers que nous étudions en **transcriptomique**. Les informations obtenues par cette science "omique" sont de double nature, d'un côté qualitative : *"quelles sont les zones du génome qui sont exprimées dans un type cellulaire donnée"* et quantitatives *"dans quelles mesures ces zones sont exprimées dans la cellule"*.

Au fil des années, de nombreuses technologies ont permis *d'explorer le transcriptome* avec de plus en plus de précision et de sensibilité. Récemment et encore aujourd'hui la technique de prédilection était **les puces à ADN** (ou microarrays en anglais). Ces dernières sont des supports sur lesquels des brins de cDNA issus d'ARN extrait d'une cellule sont hybridés avec des séquences connues et encodées sur la puce. Il était alors possible de mesurer le niveau d'expression des différentes séquences hybridées sur la plaque. Cette méthode possède l'avantage de fournir une information précise car cette technologie est très bien maîtrisée mais contrairement aux méthodes "ouvertes", les puces à ADN ne permettent de s'intéresser qu'à des ARN dont on connait la séquence.

Depuis moins de dix ans, de nouvelles technologies ont fait leur apparition, **les séquençeurs haut-débits** (SHD). Ces machines représentent une véritable prouesse technologique et permettent de séquencer des millions de fragments d'ADN en quelques

jours seulement, pour un coût relativement faible et en constante diminution. Le revers de cette technologie est que les millions de séquences produites (les *reads*) sont très courtes (50 à 200pb) par rapport à la précédente méthode de séquençage (Sanger).

Dans le cas de l'étude du transcriptome, les SHD peuvent être utilisés après avoir synthétiser des cDNA depuis les ARN de la cellule. Cette technologie se nomme le **RNA-Sequencing** (abrégé RNA-Seq). La technologie RNA-Seq permet d'accéder à la quasi-totalité du transcriptome de la cellule mais celui-ci est morcelé en des million de courts fragments. Afin d'exploiter les *reads* produits par le RNA-Seq, une des approche classique, dans le cas où un génome de référence existe, consiste à positionner les *reads* sur ce texte de référence, on appelle cette opération le *mapping*. Une fois cette étape réalisée de nombreuses études peuvent être menées comme : l'identification de polymorphismes, la quantification des ARN à l'aide d'un fichier d'annotation, des annalyses différentielles entre différents transcriptomes...

Il subsiste cependant une question que le biologiste est en droit de se poser : *"Quels était les ARN présents dans la cellule avant que ces derniers ne soient fragmentés pour réaliser le séquençage ?"*. En effet, le séquençage nous fournit une mutitude de pièces d'un immense et complexe puzzle et il est primordial d'assembler ce puzzle pour en extraire l'information biologique. Plusieurs approches ont été proposées, à ce jour, afin de résoudre ce problème d'assemblage qui ne reste pas moins une question ouverte. Les différentes approches proposées actuellement peuvent se distinguer en deux catégories :

1. **les méthodes de reconstructions guidées par génome** qui utilisent dans un premier temps un génome de référence, afin de correctement positionner les reads pour ensuite réaliser la reconstruction des transcrits.
2. **les méthodes reconstructions de novo**, qui ne nécessitent pas un génome de référence mais qui utilisent les propriétés de chevauchement des reads pour réaliser la reconstruction.

Ces deux approches ont chacune leurs avantages, les approches *de novo* sont indépendantes d'un génome de référence et de l'étape d'alignement nécessaire pour replacer les reads sur cette séquence de référence. De ce fait, ces méthodes éliminent les biais liés à l'algorithme utilisé pour réaliser l'alignement. De plus, cette indépendance vis-à-vis d'un génome de référence permet d'identifier des transcrits dans les zones manquantes du génome assemblé, ou provenant de transcrits chimériques. C'est également la seule façon d'assembler le transcriptome lorsqu'il n'existe pas de génome de référence. D'un autre côté les approches guidées par génomes sont plus performantes d'un point de

vue des ressources utilisées (CPU et mémoire) et de la précision des résultats obtenus.

Dans le cadre de ce stage, j'ai été amené à travailler sur une nouvelle stratégie d'assemblage qui se distingue des approches précédentes sur plusieurs points. Tout d'abord la partie ""de novo"" de l'assemblage se base sur une structure de donnée permettant d'indexer une collection de reads sur leur $k-facteurs$ ou $k-mers$. Notre méthode, *RASTack*, permet également le développement d'une approche intégrée permettant de coupler plusieurs sources d'information dans le processus de reconstruction.

Dans un premier temps, nous allons introduire le contexte biologique et informatique de ce projet. Dans un deuxième temps, nous nous intéresserons aux précédentes méthodes proposées pour résoudre le problème d'assemblage du RNA-Seq. Nous poursuivrons par la présentation de *RASTAck*, la méthode d'assemblage qui a été l'objet d'étude de ce stage. Enfin nous nous intéresserons aux résultats de ce stage avant de discuter des avantages, inconvénients et perspectives de ce travail.

Chapitre 1

Contexte

Sommaire

1.1 Laboratoire d'accueil : l'Institut de Recherche en Biothérapie

Le laboratoire d'accueil de mon stage, *l'**I**nstitut de **R**echerches en **B**iothérapie* (IRB), est né de l'association entre l'*Institut National de la Santé et de la Recherche Médicale* (Inserm), le *Centre Hospitalier Régional Universitaire de Montpellier* (CHRU Montpellier) et *l'Université Montpellier I* (UM1). Les locaux de l'institut se trouvent sur le campus de l'hopital Saint-Éloi à Montpellier non loin de *l'Université Montpellier 2* (UM2).

L'objectif de l'IRB est de **développer la médecine regénératrice** afin de réparer des tissus ou organes endommagés notamment grâce à l'emploi de cellules souches. Des équipes de recherche, dont celle que j'ai rejoint, sont également spécialisées dans l'étude du cancer et la découverte de biomarqueurs [1] permettant le diagnostic, le suivi, ou le pronostic de pathologies.

1. Un biomarqueur est une caractéristique biologique mesurable liée à un processus normal ou non.

L'institut accueille 110 personnes dont des chercheurs, des techniciens, des doctorants et des post-doctorants. Le laboratoire associe à la fois des équipes de recherche Inserm et CHRU, et héberge également des sociétés privées dans ses locaux. C'est un environnement propice à la collaboration entre des équipes de recherche fondamentale, des cliniciens ainsi que des entreprises. La proximité de ces différents corps de métier est particulièrement favorable à la recherche médicale et à sa mise en application directe.

Pendant ce stage j'ai rejoins l'équipe GET, **G**roupe d'**É**tude des **T**ranscriptomes, dirigée par Thérèse Commes et qui fait partie de l'unité INSERM (U1040) de l'IRB. L'équipe s'intéresse à l'étude des transcriptomes de cellules cancéreuses avec pour objectif l'identification de nouveaux biomarqueurs de la maladie. Depuis plusieurs années l'équipe GET a fait le pari que les nouvelles technologies de séquençage deviendraient la référence pour l'étude du transcriptome. Dans ce cadre, des solution bioinformatiques comprenant des structures de données [1], des algorithmes de traitement des données [2] ainsi que des pipelines d'analyse [3] ont été proposés par l'équipe afin de permettre l'analyse de ces nouvelles données.

1.2 La bioinformatique, une science pluridisciplinaire

Le mot bioinformatique est un terme relativement récent, puisqu'il n'est apparu pour la première fois qu'en 1970 dans une publication scientifique.

La bioinformatique n'est pas une science à proprement parler, mais plutôt un champ de recherche au sein duquel collaborent des scientifiques issus de différents horizons afin de travailler dans une perspective commune. La bioinformatique réunit à la fois des biologistes, des médecins, des informaticiens et des mathématiciens. Ces scientifiques collaborent afin de répondre à des questions de recherche auxquelles les biologistes seuls ne sont pas en mesure de traiter.

Les questions qui touchent la bioinformatique sont variées, mais elles ont toutes en commun la nécessité d'un support informatique conséquent. Parmi ces questions on peut citer plusieurs grands domaines comme l'analyse du génome, la modélisation de l'évolution d'une population, la modélisation moléculaire, l'analyse d'image ou encore la reconstruction d'arbres phylogéniques.

1.2.1 La bioinformatique au service de la génétique

Le pan de la bioinformatique auquel je me suis intéressé durant ce stage est le traitement des séquences génétiques issues du séquençage des transcriptomes.

Récemment, les progrès technologiques ont été particulièrement importants dans ce domaine, avec la mise au point de machines capables de séquencer un transcriptome ou un génome en une durée allant de quelques heures à quelques jours. Ces machines s'appellent des séquenceurs haut-débit.

Séquençage Extraction de l'information génétique contenue dans nos cellules pour en déterminer la séquence et la stocker sous forme numérique (par exemple sous forme d'un texte).

Une analogie pour expliquer le séquençage pourrait être la comparaison avec la numérisation de livres, à la différence qu'il est bien plus facile d'extraire l'information d'un livre que celle contenue dans chacune de nos cellules.

Un des grands défis de la bioinformatique est de développer des structures de données et des algorithmes permettant de traiter, d'analyser et d'archiver les données obtenues par le séquençage.

Ces méthodes reposent notamment sur l'algorithmique du texte et doivent être particulièrement optimisées afin de pouvoir s'exécuter dans un temps raisonnable en raison de la grande quantité de données produite par les séquenceurs haut-débit.

Pour donner un ordre de grandeur, si on représente le génome humain sous la forme d'un texte dont les lettres sont les nucléotides (ATGC), il comprendrait 3,4 milliards de caractères. Cependant les technologies actuelles séquencent aléatoirement le génome par courts fragments et il faut donc séquencer 30 fois plus de paires de bases pour obtenir une bonne couverture totale du génome. Pour un génome humain, cela représente plus de $\cong$ 100 milliards de nucléotides séquencés.

1.3 Transcriptomique

Depuis la découverte de l'ADN, le génome a souvent été considéré comme l'élément de réponse à toutes les questions posées par la biologie. Cependant, depuis que l'on connaît les séquences complètes du génome de différents êtres-vivants - dont l'Homme

- on s'aperçoit que cette information ne permet pas d'expliquer toutes les différences interspécifiques. Ainsi, même si l'homme et la souris partagent **90%** de leurs gènes, cela ne suffit pas à expliquer leurs différences.

Afin d'expliquer ce phénomène, on peut s'intéressser au mécanisme de transcription de l'ADN. La transcription transforme une partie du message génétique encodé par les molécules d'ADN et isolé dans le noyau en une autre forme chimique, l'ARN, afin que l'information puisse transiter vers le cytoplasme de la cellule et jouer son rôle biologique. Certain de ces ARN sont nommés ARN messagers (ou ARNm) car une fois dans le cytoplase ils vont être traduits en protéines par les ribosomes. D'autres ARN appellés "ARN non codants" sont biologiquement actifs sans processus de traduction, et représentent une part importante des ARN présents dans la cellule. Dans la suite de ce rapport on nommera les ARN, codants ou non-codants, **transcrits**.

On peut maintenant définir la transcriptomique comme l'étude de l'ensemble des transcrits présent dans une cellule à un moment donné. Le message du transcritpome est à la fois qualitatif : *"Quelles sont les portions du génome qui sont transcrites ?"* et quantitatif : *"Combien de copies d'un transcrit donné sont-elles présentes dans la cellule à un instant T ?"*.

Le transcriptome permet alors de répondre à la question suivante :

"Pourquoi les cellules du foie et de l'épiderme, qui ont pourtant le même patrimoine génétique dans leurs noyaux, ne se comportent pas de la même façon ?"

Chez les eucaryotes, la transcription est soumise au mécanisme d'*épissage* (voir figure 1.1). La séquence génomique est dans un premier temps recopiée de façon linéaire afin de former les *"ARN pré-messagers"*. Dans un second temps cette première molécule subit un ensemble de transformations qui vont entrainer la supression de certaines parties de sa séquence pour former *l'ARN mature*, on appelle ce processus *l'épissage*. On peut alors représenter le gène comme une structure linéaire, découpée en segments alternativement étiquettés "exons" et "introns". Les "exons" sont les portions de la séquence du gène qui seront conservées dans "l'ARN mature", les "introns" sont les portions de la séquence qui sont supprimées de "l'ARN pre-messager". Ce mécanisme biologique permet alors l'ajout d'une nouvelle couche de complexité, nommé *épissage alternatif*, qui va permettre de produire à partir d'un même gène différentes protéines dites "isoformes" (sur la figure 1.1 deux isoformes de protéine sont traduites depuis un même gène). Le mécanisme d'épissage alternatif consiste à suprimer alternativement certaines portions du gène afin de former plusieurs variants de transcritption qui donneront lieu à la traduction

de protéines différentes.

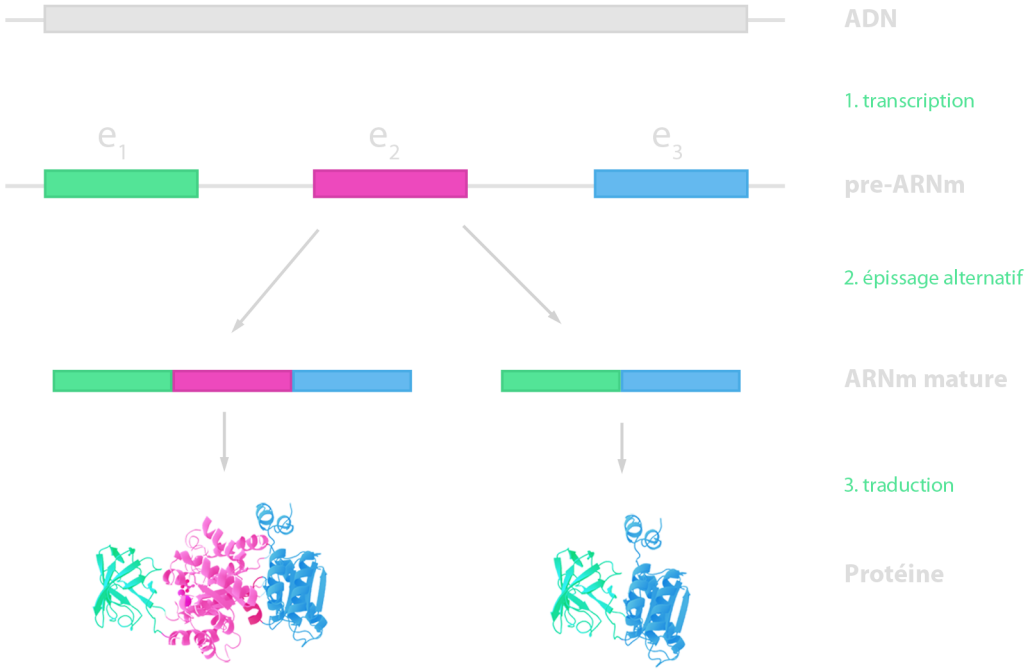

FIGURE 1.1 – Épissage alternatif d'un gènes, depuis la transcription jusqu'à la traduction.

Plusieurs phénomènes peuvent se produire lors de l'épissage alternatif d'un gène (voir figure 1.2). Le phénomène "d'exon skipping" permet de former un transcrit alternatif en supprimmant un ou plusieurs exon(s) de l'ARNm mature. D'autre phénomènes comme les variants de site donneur et receveur d'épissage peremttent de faire varier les bornes d'un exon. Il est également possible d'observer des rétentions d'introns par lesquelles une séquence considérée comme "non-codante" est conservée dans l'ARNm mature.

Ce mécanisme complexe permet d'expliquer la diversité biologique et la complexité des organismes eucaryotes. Nous allons également voir dans la suite de ce rapport que l'épissage alternatif représente un grande difficulté lors de l'assemblage des transcrits.

1.4 Séquençage Haut-débit : RNA-Sequencing

Comme nous l'avons déjà évoqué à plusieurs reprises, depuis la découverte de l'ADN, les biologistes ont cherché à développer des technologies permettant de lire le code génétique, et ce le plus facilement et le plus rapidement possible. Nous allons

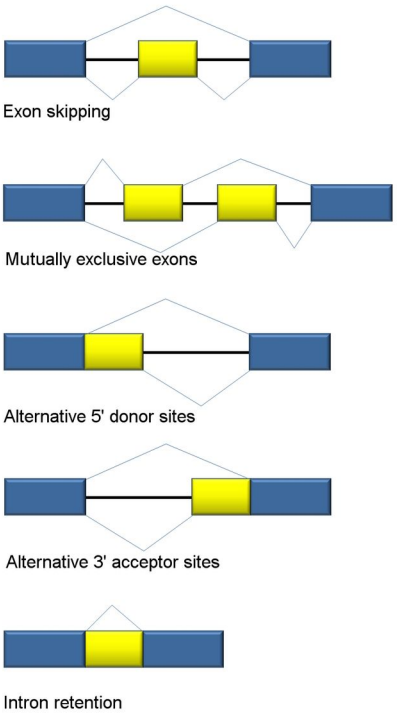

FIGURE 1.2 – Classification des différents types “classique” d’épissage alternatif. Crédit photo [4]

nous intéresser ici aux Séquenceurs Haut Débit (ou SHD) qui constituent la nouvelle génération de technologies capables de répondre à ce problème.

1.4.1 Historique du séquençage

Pendant plus de 30 ans, la **méthode Sanger** a permis de décoder le message génétique des organismes vivants. Cette méthode a notamment permis de réaliser *le décryptage du génome humain* au cours des années 2000. Il aura fallu pour cela la collaboration de plusieurs grands laboratoires internationaux pendant presque **13 ans**, avec un budget de deux milliards de dollars US.

Cependant, en 2007, de nouvelles machines sont apparues sur le marché avec un débit 50 à 1 000 fois supérieur. Ces machines se nomment les **Séquenceurs Haut Débit** ou Next Generation Sequencing en anglais (NGS). Ces nouveaux séquenceurs, dits “haut-débit”, ont permis de révolutionner les analyses génomiques et transcriptomiques. Ces machines ont cependant un coût très important (plusieurs centaines de milliers d’euros) mais ces prix sont en baisse et le champ d’application de ces appareils est très étendu.

Aujourd'hui, Illumina propose de séquencer un génome pour moins de 1000 dollars.

1.4.2 Séquençage haut-débit

Les séquenceurs haut débit, comme l'Illumina MiSeq de la figure 1.3 reposent sur des technologies légèrement différentes (lecture laser, capteur photographique...), mais leur fonctionnement est similaire. Ces machines isolent les différents fragments d'ADN contenus dans l'échantillon sur un support où ils vont être chacuns amplifiés. La phase suivante consiste à lire le signal des différents fragments amplifiés. Dans le cas d'illumina c'est un capteur photographique qui va lire le fragment base par base à l'aide de fluorochromes se fixant sur les fragments et permettant d'attribuer à chacune des bases A, T, G, C une couleur différente. Un analyse d'image est ensuite effectuée afin de déterminer pour chaque point lumineux du support, représentant un fragment d'ADN, quelle est la base qui vient d'être lue.

FIGURE 1.3 – Séquenceur haut-débit Illumina.

Ces machines sont capables de lire en parrallèle des millions de fragments de cDNA, et d'enregistrer leur séquence, appelée *read*, dans un fichier FASTQ [1].

Un read est le nom que l'on donne à un fragment ADN séquencé par un SHD. Les reads sont généralement longs d'une centaine de paires de bases, mais cela peut varier suivant la technologie et la génération de séquenceur utilisés.

1. Le format FASTQ est un format de fichier texte permettant de stocker à la fois des séquences biologiques (uniquement nucléiques) et les scores de qualité associés.

1.4.3 RNA-seq

Le **RNA-seq** (RNA-Sequencing) est l'une des applications des séquenceurs haut débit qui rencontrent le plus de succès. L'objectif du RNA-Seq est de séquencer un ensemble de transcrits extraits depuis un échantillon biologique.

La méthode du RNA-seq (voir figure 1.4) repose sur un protocole qui intègre la reverse transcritpion des mRNA en cDNA leur fragmentation avant les étapes de séquençage. Des adaptateurs sont ajoutés aux extrémités des fragments afin que ces derniers puissent se fixer sur le support du séquenceur. On appelle cette opération la "préparation la bibliothèque RNA-SEQ" (ou library en anglais). Cette bibliothèque est ensuite déposée sur un support du séquenceur qui va alors générer des millions de lectures (appelées reads), en une étape appelée "run".

Il existe plusieurs variantes des protocoles RNA-Seq permettant d'obtenir des informations supplémentaires sur les reads : le protocole **orienté** et le protocole **paired-end reads**.

Le protocole orienté (ou stranded en anglais), permet de connaître **l'orientation d'un read**, afin de savoir à partir de quel brin de l'ADN a été transcrit l'ARN à l'origine de ce read.

Le protocole Paired-end reads (également appelé PE), consiste à séquencer les **deux extrémités d'un fragment** afin de former des paires de reads éloignées d'une centaine de pb environ.

1.4.4 Limites et futur des SHD

La nouvelle génération de séquenceurs promet de révolutionner de nombreux domaines de la biologie grâce à l'accès quasi-immédiat aux données génétiques.

Il faut cependant noter que tous les SHD produisent des erreurs de séquence et qu'il est particulièrement important de développer des méthodes informatiques et statistiques permettant de contrôler cet aléa. De plus, la faible taille des fragments séquencés est également un problème dans le cadre de nombreuses applications, comme c'est le cas pour la reconstruction des transcrits. Enfin, les différentes manipulations biologiques réalisées pour obtenir la bibliothèque RNA-Seq sont responsables d'artefacts de séquence.

Le futur des séquenceurs, semble très prometteur. En effet, la taille des séquences

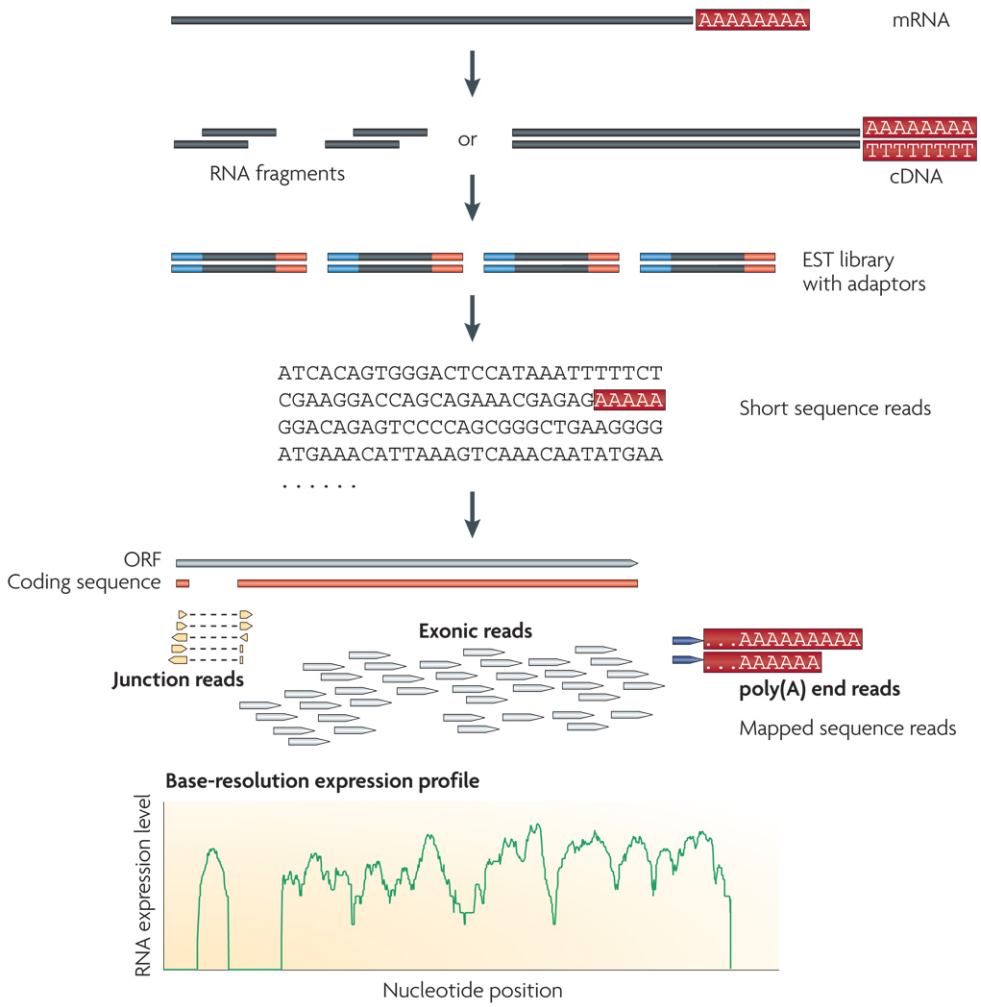

FIGURE 1.4 – Le protocole de séquençage haut-débit du transcriptome : **RNA-Sequencing**.

obtenues ne cesse de croître, de même que la volumétrie des reads générés à chaque **“run”**.

De nouveaux séquenceurs “haut-débit” pouvant générer des “reads” de grande taille sont actuellement en train de voir le jour, comme par exemple le *Pacific Biosciences*. Cependant, ces appareils sont très couteux et génèrent de nombreuses erreurs de séquence.

Chapitre 2

Etat de l'art

Sommaire

Nous allons voir dans ce chapitre, quelles ont été les méthodes et les structures de données qui ont été proposées, afin de répondre au problème de la reconstruction des transcrits à partir d'une collection de reads issue d'une expérience RNA-Seq. Comme nous l'avons vu dans le chapitre précédent, la complexité du transcriptome Eucaryote ainsi que celle des données haut-débit représente un véritable défi pour l'assemblage des reads RNA-Seq.

Comme indiqué dans l'introduction, on peut regrouper les différentes méthodes proposées à ce jour en deux catégories :

— les approches guidées par génome
— les approches par assemblage "de novo"

Ces deux types d'approches sont résumées dans la figure 2.1. Dans le cas de la reconstruction guidée par génome (à gauche de l'image) les reads sont d'abord positionnés sur le génome de référence grâce à des algorithmes de mapping, afin de détecter les exons des gènes et les jonctions d'épissage nécessaires pour proposer des modèles de transcrits. Dans le cas de l'assemblage "de novo", les reads sont assemblés les uns avec les autres sur un critère de ressemblance entre leurs séquences. Enfin, les transcrits ainsi assemblés sont

positionnés sur le génome de référence, afin de mettre en évidence la structure "exonique" de ces derniers.

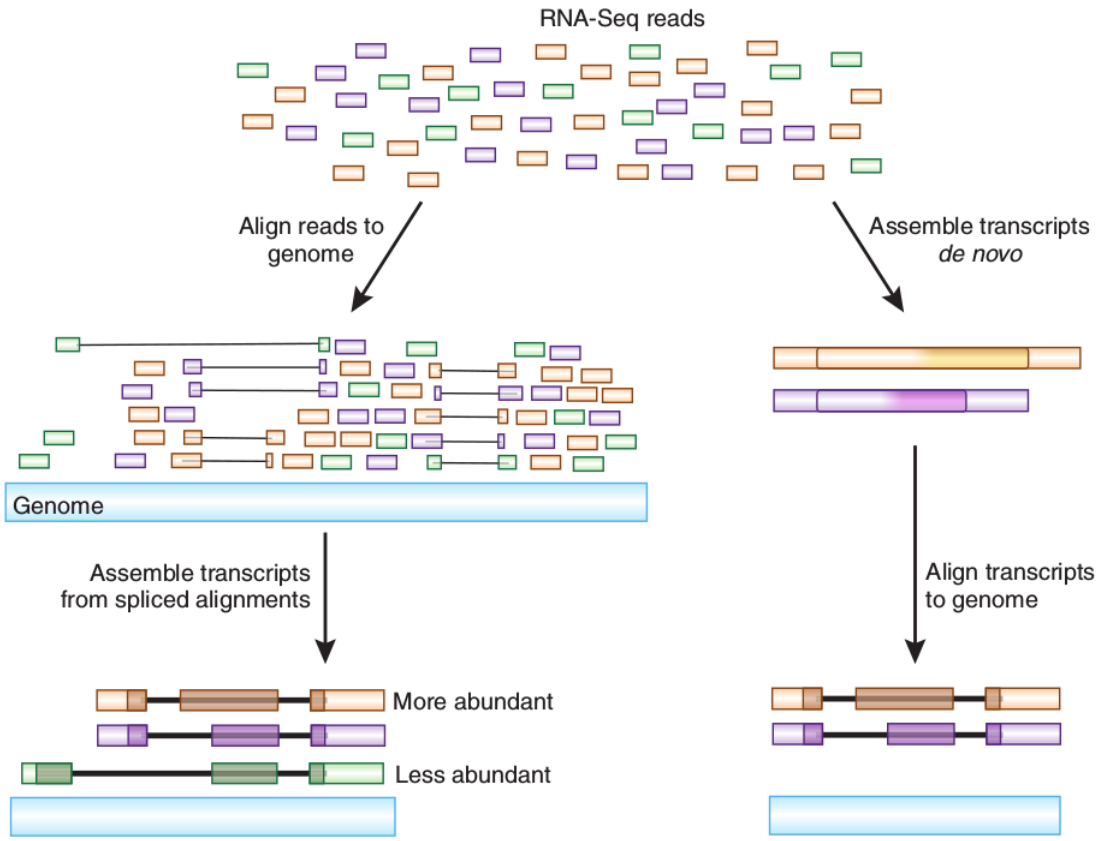

FIGURE 2.1 – Les deux grandes approches de la reconstruction des transcrits à partir de données RNA-Seq. Crédit photo [5].

2.1 Reconstruction guidée par génome

La reconstruction guidée par génome a fait ses preuves depuis la diffusion de Tophat-Cufflinks [6] et de Scripture [7]. Ces deux outils sont très largement cités ($\approx$ 1000 citations pour Cufflinks) dans la littérature et utilisés pour l'analyse du transcriptome.

Les algorithmes de reconstruction *guidée par génome* se basent sur une première étape de mapping au cours de laquelle les reads sont alignés sur un génome de référence. Une seconde étape permet souvent d'aligner les reads qui comportent une jonction d'épissage afin d'identifier les variants d'épissage et les sites donneurs et accepteurs d'épissage. Il est alors possible d'identifier les zones du génome correspondant à un gène, puis les exons

de ce gène et enfin de les grouper sous forme de transcrits grâce aux reads qui comportent l'information de la jonction d'épissage. Une dernière étape de quantification permet alors d'estimer l'abondance des différents transcrits ainsi construits.

Récemment, deux nouvelles approches de reconstruction *guidée par génome* ont fait leur apparition, afin de combler les limites des méthodes existantes. Une des principales limites de Cufflinks est de construire l'ensemble minimum de transcrits permettant d'expliquer les jonctions d'épissage. Ces deux méthodes proposent notamment de corriger cette limite. La première, CLASS [8], propose d'utiliser un graphe de contraintes afin de mieux modéliser les variants d'épissage. La seconde, GRIT [9], propose une méthode intégrée permettant de rassembler l'information de plusieurs technologies de séquençage afin de produire de meilleurs modèles de transcrits.

2.2 Assemblage de novo

Les méthodes de reconstruction dites *de novo* [10, 11, 12] cherchent à reconstruire le transcriptome avec pour seule information, la séquence des reads produits par une expérience de RNA-Sequencing. Ces méthodes sont souvent utilisées pour l'analyse RNA-Seq d'espèces non modèles où aucun génome correctement assemblé n'est disponible.

Les méthodes de reconstruction *de novo* sont proches des méthodes d'assemblage de génome car elles se basent sur les mêmes structures, afin d'organiser l'information des reads en vue de les assembler.

2.3 Approches combinées

Des travaux plus récents proposent le développement de méthodes hybrides entre guidage par génome et assemblage de novo. Une étude de 2013 [13] a quantifié l'intérêt de combiner les résultats des approches "guidées par génome" et "de novo" sur un même jeu de données, afin de combiner les forces et limiter les faiblesses de chacune à reconstruire certains transcrits. Une seconde étude [14] apporte des conclusions similaires sur le fait de combiner les résultats de différentes approches et différents logiciels, afin d'augmenter l'assemblage produit. L'étude ouvre également la perspective de développer une approche intégrée qui permettrait de combiner les approches actuelles pendant l'étape

d'assemblage.

2.4 Assemblage "de novo" et structures de données

Afin d'assembler les reads RNA-Seq sur la base de la ressemblance de leur séquence, des structures de données permettant de faciliter ce travail on été proposées. Ces structures ont pour la plupart été développées dans le cadre du problème d'assemblage des génomes.

2.4.1 Overlap graph et string graph

Un graphe d'overlap (voir figure 2.2) est un graphe dont les noeuds sont des reads et les arrêtes des relations de chevauchement suffixe/préfixe supérieure à k entre deux reads.

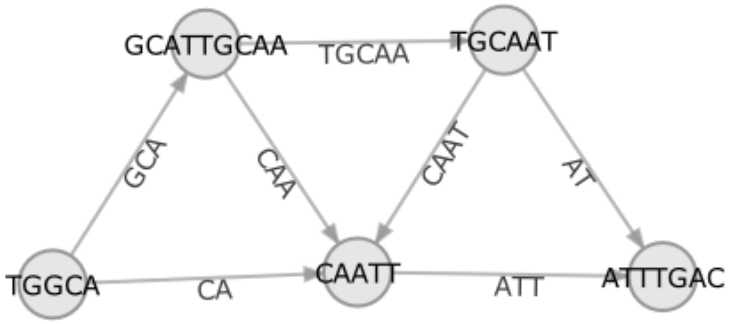

FIGURE 2.2 – Overlap graphe constitué à partir des mots (GCATTGCAA,TGCAAT,TGGCA,CAATT,ATTTGAC) avec une relation de chevauchement minimum $k > 1$.

Cette structure a été particulièrement utilisée lors de l'assemblage des génomes avant d'être remplacée par les graphes de de Bruijn. Cette structure possède l'avantage de conserver la séquence complète des reads, cependant, elle est très gourmande en ressources. D'une part, calculer les relations de chevauchement entre tous les reads représente un traitement très lourd. D'autre part, l'impact mémoire de cette structure est très important et peu adapté aux données haut-débit. En effet, le graphe comporte beaucoup de redondances dues à la conservation de la séquence complète des reads dans chacun des noeuds. Il existe également un grand nombre d'arrêtes reliant les reads les uns aux autre.

Récemment, une nouvelle structure, le "string graph", propose de diminuer la taille de l'overlap graph, en supprimant la redondance de l'overlap graph. Cette structure est associée à un algorithme [15] permettant sa construction depuis un FM-index réduisant ainsi le temps de calcul de manière significative.

Cependant, cette structure n'est pas complètement adaptée au problème de la reconstruction des transcrits car le mécanisme d'épissage alternatif n'est pas facile à prendre en compte.

2.4.2 Approches par $k-mer$

Les deux structures suivantes ont en commun une approche par *k-mer*. L'approche par *k-mer* consiste à découper les reads en sous-chaines de longueur k, appelées *k-mers*, qui servent alors d'unités atomiques pour accéder à l'information.

Pour une longueur k donnée il existe, pour l'ADN, 4^k combinaisons possibles. Ainsi, pour un k suffisamment élevé, si un *k-mer* apparait sur des reads distincts, la probabilité que ceux-ci soient issus d'une même zone de l'ADN est grande. Cette longueur doit également être suffisamment grande pour qu'un *k-mer* généré aléatoirement ne trouve pas de correspondance dans le texte de référence, autrement dit qu'il n'y ait pas de collision.

L'approche par *k-mer* consiste à déterminer pour une espèce donnée une longueur k optimale. Cette approche a démontré son efficacité [16] et il existe de nombreux outils permettant de déterminer la longueur k optimale [17] pour une espèce donnée. Par exemple, la longueur k optimale pour le génome humain est 21. Pour chacune des structures présentées dans ce rapport, le paramètre k est très important. D'une part, il va déterminer la taille que la structure va prendre en mémoire. D'autre part, il définira le temps de construction et de requêtage. Mais, le paramètre k va également être important d'un point de vue biologique pour la pertinence de la donnée.

2.4.3 Graph de de Bruijn

Le graphe de de Bruijn (voir figure 2.3) est actuellement la structure de référence pour l'assemblage des reads RNA-Seq.

Un graphe de de Bruijn est un graphe orienté qui permet de représenter les chevauchements de longueur $n-1$ entre tous les mots de longueur n sur un alphabet donné.

[18]

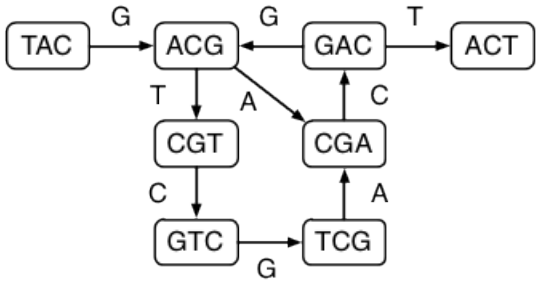

FIGURE 2.3 – Graphe de de Bruijn construit sur l'alphabet génétique avec $(k = 2)$ et sur la séquence *TACGACGTCGACT*.

L'emploi de cette structure a été suggéré par [19] pour l'assemblage des reads, car l'empreinte mémoire d'un *DBG* est bien inférieure à celle des *graphes de chevauchements de reads* précédemment utilisés par les assembleurs. Une fois le *DBG* construit, il est alors possible de le parcourir afin d'identifier des chemins correspondants aux différents transcrits présents dans la cellule au moment du séquençage.

Cette structure possède un réel avantage : sa taille. En effet, le découpage des reads en $k-mers$ permet de supprimer la redondance présente dans les "overlap graphs". Le désavantage du graphe de de Bruijn est la perte de l'information sur la séquence complète des reads. Une fois le graphe construit, il n'est alors plus possible de savoir si la séquence d'un chemin du graphe a été créée grâce à un read. Cela pose de grandes difficultés pour l'assemblage de transcrits qui contiennent des séquences répétées dans le génome.

2.4.4 Les GkArrays

La dernière structure qui sera présentée ici est les GkArrays[1]. Cette structure n'a jamais été utilisée pour l'assemblage des reads RNA-Seq mais possède un ensemble de propriétés permettant de réaliser un tel traitement.

La structure des GkArrays permet d'indexer l'ensemble des *k-mers* des reads et offre via des requêtes sur les *k-mers* l'avantage de pouvoir remonter jusqu'aux reads dont ils sont issus. Cette structure trouve de nombreuses applications notamment dans les problématiques d'assemblage ou de *clustering* de *reads* qui vise à identifier un ensemble de *reads* chevauchants (ie. qui partagent un ou plusieurs k-mer(s)).

Les GkArrays possèdent donc des capacités intéressantes pour résoudre les problématiques d'assemblage en permettant le regroupement de reads sur la base d'un $k-mer$ partagé. Nous allons d'ailleurs voir dans le chapitre suivant comment les GkArrays vont pouvoir donner naissance à une nouvelle méthode d'assemblage du RNA-Seq.

Chapitre 3

Méthode

Sommaire

Ce chapitre est consacré à une nouvelle méthode d'assemblage des données RNA-Seq qui a été l'objet d'étude de mon stage. Cette méthode nommé *RASTAck* (**R**ead **A**ssembly with a **STAck**) se place à l'échelle du "read" et utilise la notion de $k-mer$ (ou $k-facteur$), maintes fois décrite dans la littérature [16], afin de rapprocher des reads partageant une même sous-séquence de longueur k. Cette approche novatrice est rendue possible grâce à une structure de données [1] qui indexe l'ensemble des $k-mers$ des reads en mémoire et propose des requêtes sophistiquées afin d'accéder à l'information du transcriptome séquencé depuis un $k-mer$.

Nous allons voir dans ce chapitre les différents concepts, structures et algorithmes qui ont été développés dans le cadre de ce stage afin de créer une nouvelle méthode d'assemblage des reads à partir d'une expérience RNA-Seq.

3.1 Philosophie d'assemblage

Cette première section présente les objectifs de notre méthode d'assemblage ainsi que les principes sur lesquels est ancrée la méthode.

3.1.1 Principe : *"les reads disent la vérité"*

La philosophie de *RASTAck* est simple, elle part de l'hypothèse que **les reads disent la *vérité***. En effet, chaque séquence générée par le séquenceur haut-débit porte une information du transcriptome qui a été séquencé. Les reads sont autant de pièces d'un immense puzzle, à la différence que le puzzle du RNA-Seq est constitué de milliers d'images (les transcrits) parfois entrelacées les unes avec les autres (transcrits alternatifs, pseudogènes...), et dont certaines pièces sont manquantes ou altérées.

Dans la méthode présentée ici, chaque read apporte une information précieuse sur le transcriptome, et, est donc pris en compte dans le processus de reconstruction de la première à la dernière base. Il existe également certains reads qui ne disent pas la "vérité", conséquence d'une production d'artefacts lors de la préparation de la librairie RNA-Seq ou d'erreurs lors du séquençage des fragments de cDNA. Cependant, nous allons voir que ces derniers peuvent être repérés et écartés afin de ne pas brouiller le signal du transcriptome.

Il est important de souligner que la méthode d'assemblage proposée ici se place **à l'échelle du read**. Il existe de nombreux avantages à utiliser le read comme unité d'assemblage, d'une part il n'y a aucune perte d'information dans le signal issu du séquençage, d'autre part cela permet le développement **d'approches intégrées** couplant l'information sur la séquence des reads ainsi que d'autre données telles que l'alignement des reads sur un génome de référence. À la différence des méthodes basées sur *les graphes de de Bruijn* qui découpent les reads en fragments chevauchants de longueur k et qui perdent donc l'information de la séquence complète, notre méthode est en capacité d'utiliser la séquence complète d'un read et ce avec un temps de traitement très raisonnable grâce à notre structure d'indexation, les GkArrays. Cela représente un avantage essentiel de notre méthode qui est donc plus robuste aux séquences répétées ou dupliquées dans le génome.

3.1.2 Objectif : *"la reconstruction des meta-reads"*

Le premier objectif fixé par la méthode de *RASTack* n'est pas la reconstruction complète des transcrits, mais la reconstruction d'éléments d'une échelle inférieure que nous nommons les *meta-reads*.

Un *meta-read* est un assemblage de reads, ou de facteurs de reads, qui partagent la même histoire.

Une *histoire* est une séquence du génome, continue ou non, qui est partagée dans son intégralité par les mêmes transcrits de la première à la dernière base.

Plus formellement, chaque transcrit est partitionné en *meta-reads* qui ne peuvent se chevaucher. À l'inverse, il peut arriver que des transcrits différents soient chevauchants et partagent donc une sous-séquence commune, comme c'est le cas dans un épissage de type "alternative donor/acceptor site" (voir la partie *B* de la figure 3.1). Dans ce cas, plusieurs *meta-reads* seront créés afin de partitionner les différentes *histoires* observées ; celles étant communes à différents transcrits et celles propre à chacun. Des approches similaires aux *meta-reads* ont déjà été décrites dans la littérature [20] pour d'autres applications à partir de données RNA-Seq.

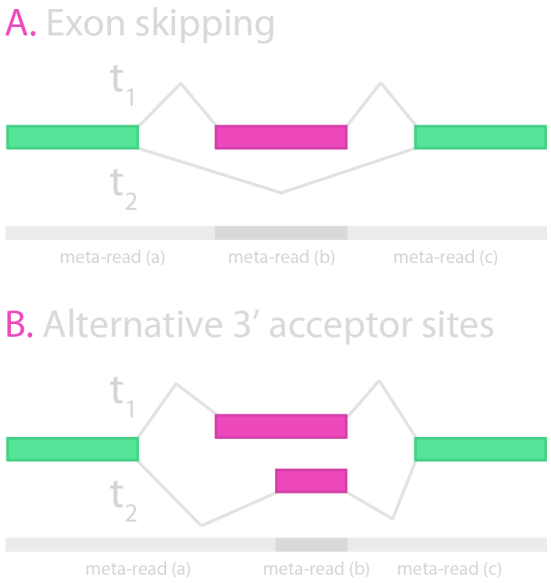

FIGURE 3.1 – Cette figure présente la notion de *meta-reads* en fonction de différents événements d'épissage alternatif pouvant se produire lors de la transcription d'un gène. Les différents *meta-reads* ainsi construits sont projetés sur la région génomique correspondante et sont distingués par leur couleur (en niveau de gris) et leur notation (meta-read({a,b,c}).

La reconstruction des *meta-reads* présente un double-intérêt :

1. Premièrement, il s'agit d'une première étape vers l'assemblage complet des transcrits. En effet, si l'on suppose que les *meta-reads* on été correctement assemblés il faut alors les combiner afin de former les différents transcrits. Cette étape est déjà prise en compte par différents logiciels [8] de reconstruction "guidée par génome"

qui, après avoir détecté les bornes des exons et les jonctions d'épissage forment toutes les combinaisons potentielles de transcrits alternatifs. D'autre logiciels utilisent ensuite des méthodes de "maximum de vraisemblance" [6], ou des approches sans alignement [21], afin d'inférer l'abondance des différents isoformes. Il serait donc possible d'utiliser ces mêmes méthodes afin d'assembler nos *meta-reads* en transcrits puis les quantifier.

2. Deuxièmement, les *meta-reads* fournissent un niveau d'information très suffisant pour de nombreuses analyses telles que les analyses différentielles. En effet, les *meta-reads* pourraient simplifier les méthodes de quantification par comptage des reads qui posent de nombreux problèmes pour assigner un read à un transcrit lorsque celui-ci est aligné sur un exon partagé par plusieurs transcrits. Une étude ultérieure permettrait de démontrer l'intérêt d'une analyse différentielle basée sur un comptage par *meta-reads*.

Un second objectif de notre méthode est de pouvoir permettre le croisement de multiples sources d'information pendant le processus de reconstruction, afin de pouvoir lever une ambiguïté où détecter des événements qui ne sont visibles que depuis une source d'information particulière. Dans ce cadre, nous avons décidé d'intégrer les informations sur l'alignement des reads afin de pouvoir distinguer deux reads qui partagent un même $k-mer$ mais qui racontent deux *histoires* différentes. Ce rapprochement entre les méthodes "de novo" et "guidées par génome" est permis grâce à notre structure de données qui fait le lien entre un $k-mer$ et les reads qui ont une occurrence de ce dernier. En plus de l'alignement des reads, nous pouvons imaginer intégrer d'autres sources d'information telles que des données CAGE et SAGE, comme le fait la méthode CLASS[8] afin de détecter avec plus de précision les bornes des transcrits et identifier des débuts ou fins de transcription alternative.

3.2 Les concepts clef

Afin de procéder à la reconstruction des *meta-reads* depuis une collection des reads indexés par les GkArrays, plusieurs concepts ont été mis en évidence.

3.2.1 Collection de reads

Une collection de reads est un ensemble de reads indexés dans un structure de données qui va permettre d'extraire l'information sur les reads et la réponse à des requêtes sur la base d'un $k-mer$ présent dans la collection.

Dans le cas des *GkArrays*, la notion de facteur n'est pas directement liée à la séquence du facteur lui-même, mais à un couple de nombres entiers composé de l'identifiant d'un read et de la position d'un de ces $k-mer$.

De façon formelle, on peut définir une collection de reads *rc* comme une structure qui indexe une collection de N reads sur leurs facteurs de longueur k. Un *read* est un mot définit sur l'alphabet $\sigma = {}'A',{}'T',{}'G',{}'C'$. Pour chaque read indexé dans *rc* il existe un indice $i \in 0..N-1$ correspondant à l'identifiant du read et qui permet d'accéder à ces informations. Chacun des reads de la collection peut avoir une longueur variable, $m(i)$ définit donc la longueur de la séquence du read à l'indice i. Pour un read i de la collection *rc* on peut accéder à un de ces $k-mers$ depuis une position p avec $p \in 0..m(i)-k-1$.

En utilisant cette formalisation il est possible de définir un cahier des charges de la structure qui sera constitué d'un ensemble de requêtes permettant de fournir un accès aux données de la collection.

Requêtes sur les reads

- Q_a **Combien de reads** sont indexés dans la *collection* ?
  Retourne N
- Q_b Quelle est la **longueur de la séquence du read** i dans la *collection* ?
  Retourne $m(i)$
- Q_c **Quelle est la séquence du facteur** de longueur l à la position p du read i ?
  Retourne la séquence du facteur du read i définit par l'intervalle $[p..p+l-1]$.

Requêtes sur les $k-mers$

- Q_1 : **Combien de reads partagent le facteur** à la position p du read i ?
  Retourne un entier correspondant au nombre de reads.
- Q_2 : **Quels sont les reads qui partagent le facteur** à la position p du read i et **quelle est la position de ce facteur dans ces reads** ?

Retourne une liste de couple (i, p) où i est l'identifiant d'un read qui contient le $k-mer$ recherché et p est la position de ce k-mer dans le read i.

Selon l'état de l'art, les GkArrays constituent la seule structure capable de répondre à l'ensemble de ces cinq requêtes. De plus, **la nature des GkArrays et notamment le fait que la structure soit chargée en mémoire permet d'obtenir des performances très intéressantes pour un algorithme de reconstruction**. C'est pourquoi cette structure a été choisie pour implémenter l'algorithme de *RASTAck*. Il serait cependant possible de substituer les GkArrays à une structure équivalente qui aurait un impact mémoire inférieur. C'est notamment une des perspectives de ce stage (voir 5). Des efforts d'implantation ont été prévus à cet effet, afin de rendre générique l'utilisation d'une *collection des reads* dans le processus d'assemblage. Nous verrons cela dans le chapitre 4.

Il existe également certaines spécificités des données RNA-Seq, dues aux différents protocoles existants, qu'une collection de reads peut prendre en compte.

Les collections de reads orientés ou non-orientés

Comme nous l'avons vu dans la section 1.4.3, les données RNA-Seq peuvent être orientées ou non. Dans le cas des données non-orientées, une collection de reads devrait effectuer la gymnastique du "reverse-complément" afin qu'**un read séquencé sur un brin ou l'autre de l'ADN soit équivalent**. Les GkArrays sont capables d'accomplir cet exercice. De plus, cela permet dans le cas de données non orientées de réduire l'impact mémoire de la structure. En effet, une des tables qui compose les GkArrays enregistre pour chaque $k-mer$ de la collection de read son nombre d'occurence, afin de répondre à la requête Q_1. Dans le cas des données non-orientées cette table est très fortement réduite puisqu'un k-mer et son "reverse-complément" ne forme qu'une seule entrée dans la table.

Les collections de reads paired-end ou single-end

Le protocole paired-end (ou PE) permet d'obtenir depuis un fragment d'ADN un couple de reads qui sont séquencés depuis les extrémités du même fragment de cDNA. Permettre de faire le lien entre deux reads "pairés" fournit une information d'une grande importance car cela offre l'accès à une séquence plus grande que celle des reads ; même si cette dernière est partiellement incomplète. Cependant, il est possible de connaître la taille des fragments de la librairie RNA-seq, ce qui apporte une information sur la taille de la séquence non-séquencée appelé "insert-size". Comme pour le protocole orienté, les

GkArrays ont été adaptés afin de prendre en compte le protocole PE pour pouvoir passer d'un read à sa paire. Cette adaptation représente un travail que j'ai réalisé lors d'un précédent stage dans l'équipe de T.Commes à l'IRB.

Ces deux protocoles RNA-Seq peuvent s'intégrer parfaitement dans la méthode que nous proposons. Cependant, **afin de simplifier la description de la méthode, nous allons supposer ici que les données que nous utilisons sont *orientées* et en *single-end***.

3.2.2 Le profil de support d'un read

La requête Q_1 de la collection de reads permet de définir le concept de **profil de support** d'un read qui a précédemment été utilisé par l'algorithme de mapping CRAC [2], afin de séparer les erreurs de séquençage des causes biologiques.

Le profil de support (noté s) définit pour chacun des $k-mers$ d'un read son support. La notion de support d'un $k-mer$ est relative à la requête Q_1 d'une *collection de reads* qui permet d'obtenir le nombre de reads de la collection qui partagent un $k-mer$ donné. La **longueur du support s d'un read i**, notée $m(s_i)$, est définie par la longueur de sa séquence ainsi que k la longueur des $k-mers$ qui ont servi à indexer la collection de reads :

$$m(s_i) = m(i) - k + 1$$

Le profil de support permet d'obtenir *une empreinte du read* dans la collection. Étudier ses variations permet de voir de nombreux événements biologiques ou non, tels que les erreurs de séquençage, les duplications, les répétitions, les SNP ou encore les variants d'épissage.

On peut notamment attendre d'un read complètement inclus dans un *meta-read* et ne possédant pas d'erreur de séquençage d'avoir **un profil de support "stable" indiquant que le read est impliqué dans une seule *histoire*** : celle du *meta-read* en question. Cette hypothèse est vraie dans le cas ou la couverture du RNA-Seq est suffisante et que les reads sont aléatoirement répartis le long des transcrits.

Afin d'étudier le *profil de support* d'un read, il est possible d'appliquer certaines mesures statistiques basiques :

La **moyenne du *profil de support*** (noté $\bar{s}$), permet d'obtenir une idée du niveau d'expression du *meta-read* auquel le read appartient. Comme souvent en statistique, cette mesure n'est pas suffisante et doit être couplée à une mesure de dispersion, afin d'avoir une idée plus précise de la variation de la donnée. On peut donc calculer **l'écart-type du *profil de support*** (noté σ_s) afin de mesurer la stabilité ou l'instabilité du *profil de support*.

Afin de coupler ces deux valeurs et pour donner une estimation de la fiabilité d'un read à décrire une seule *histoire*, nous avons mis en place un score associé au *profil de support* d'un read. Ce score est compris entre 0 et 1 de telle façon qu'un score égal à 1 indique qu'il n'y a aucune variation dans le *profil de support* et qu'un score égal à 0 indique que la variation du support est supérieure à la moyenne elle-même, ce qui indique une grande variabilité. Le score est calculé de la façon suivante :

$$read\ score = (\frac{-1}{\bar{s} \times \sigma_s}) + 1$$

Une étude précédente [22] réalisée par un étudiant de Master 1 avait permis l'exploration d'une méthode d'assemblage basée uniquement sur le *profil de support* des reads et la concaténation de reads chevauchants. Nous allons voir dans la suite de ce chapitre des concepts permettant d'enrichir cette information afin de reconstruire les *meta-reads* de façon précise et efficace.

3.2.3 La stack : un empilement des reads

La requête Q_2 de la collection de reads permet de définir le concept de **pile**.

Une *pile* (ou stack en anglais) est un empilement de reads qui a été constitué sur la base d'un facteur commun, partagé par l'ensemble des reads de la pile. Ce facteur d'empilement sera nommé par la suite : *le facteur de référence*.

Pour chacun des reads de la *pile* on peut définir un couple d'identifiants i, p :

— $i \in [0, N-1]$, **l'indice du read** dans la collection,
qui va permettre d'obtenir des informations détaillées sur ce read
— $p \in [0, m(i)-1]$, **la position du** $k-mer$ ***de référence*** dans le read,
qui va permettre de déterminer la façon dont seront empilés les reads dans la pile.

Les reads de la pile sont superposés les uns sur les autres et alignés par rapport au *facteur de référence* qui a permis d'établir la pile (voir partie **A** et **B** de la figure 3.2).

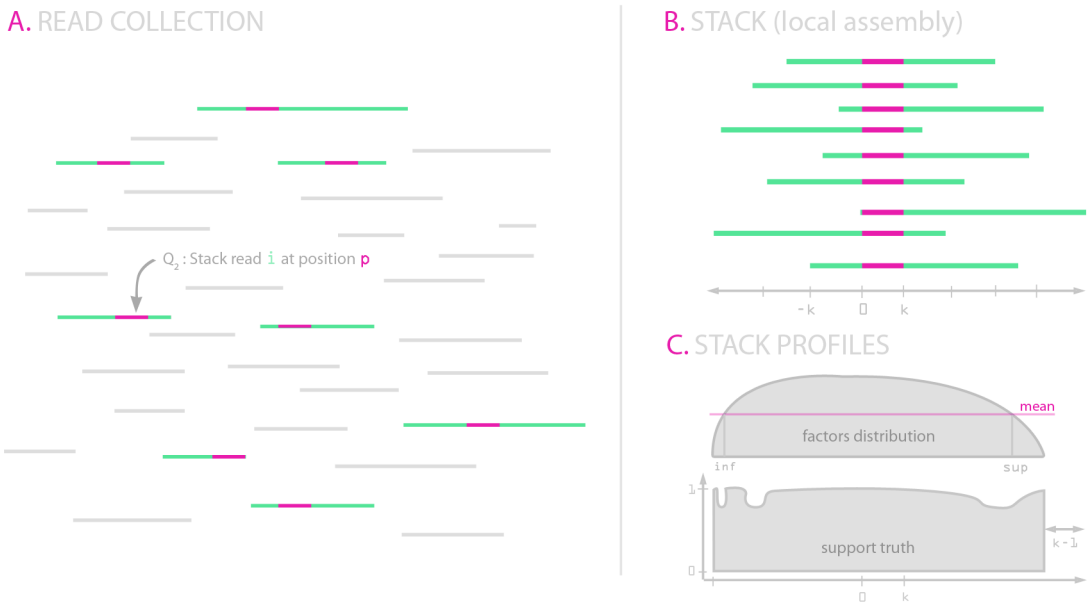

FIGURE 3.2 – La partie *A* de la figure représente une collection de reads (les segments), dont certains on été mis en évidence (couleur bleu) car ils partagent un $k-mer$ commun. La partie *B* montre comment ces reads sont empilés et alignés par rapport au facteur de référence représenté par un segment rose. Enfin, la partie *C* de la figure met en évidence le profil de la pile et le support de vérité. Le profil de la pile représente la couverture de la pile à chaque position. Le support de vérité donne une information sur la proportion de reads racontant une seule et même histoire à une position de la pile.

On peut donc représenter une pile dans un espace à une dimension, dont on projette les reads de la pile sur l'axe unique, nommé x. Ces derniers sont représentés sous la forme d'un segment qui est placé sur l'axe de sorte que la position du *facteur de référence* dans le read soit placé sur l'origine de l'axe (à savoir 0). Pour un read i dont la position du *facteur de référence* est p, le segment du read sera placé sur l'axe avec l'intervalle $[-1 \times p, m(i) - p - 1]$.

La *pile* est bornée sur l'axe x par le read aligné le plus à gauche et le read aligné le plus à droite. Soit $max(p)$, la valeur maximum de p, la position du *k-mer de référence*, pour tous les couples i, p des reads de la pile et $max(m(i) - p)$ la valeur maximum de la différence entre la taille d'un read ($m(i)$) et p pour tous les couples i, p des reads de la pile, alors on peut définir les bornes inférieures et supérieures de la *pile* sur l'axe x par l'intervalle :

$$[inf(pile), sup(pile)] = [max(p) \times -1, max(m(i) - p) - 1]$$

3.2.4 Les propriétés d'un empilement de reads

On peut définir plusieurs propriétés d'une pile :

La *hauteur maximum de la pile* est le nombre total de reads qui ont été empilés. C'est le résultat donné par la requête Q_1 de la collection des reads sur le *facteur de référence*. Cette *hauteur maximum* peut être observée à l'origine de l'axe x, la position du *facteur de référence*.

La *longueur de la pile* est définie par la distance entre la borne inférieure et supérieure de la pile : $longueur\,de\,la\,pile = sup(pile) - inf(pile)$

La *hauteur relative de la pile* définit pour une position q de la pile, avec $q \in [inf(pile), sup(pil)]$, le nombre de reads chevauchant cette position. Un read est dit chevauchant à une position q si le segment $[a, b]$ qui le représente vérifie $q \geq a \wedge q \leq b$.

Ces trois propriétés apportent à elles-seules des informations très intéressantes sur l'empilement de reads.

La *hauteur maximum de la pile* apporte une information quantitative et nous donne une indication sur le niveau d'expression du *meta-read* que l'on cherche à reconstruire.

La *longueur de la pile* apporte une information qualitative sur le *meta-read*, notamment sur les extrémités des transcrits. Prenons l'exemple d'une collection dont les reads ont une longueur constante m, la **longueur théorique maximum de la pile** est alors de de $2 \times m - k$. Cette longueur théorique est atteinte lorsqu'un premier read contient le $k-mer$ *de référence* à la fin de sa séquence, soit à la position $m-k-1$, et qu'un second read contient le $k-mer$ *de référence* au début de sa séquence, soit à la position 0. Si **l'on suppose que les reads issus du RNA-Seq sont aléatoirement distribués le long des transcrits**, il est attendu que la *hauteur maximum de la pile* soit corrélée à la *longueur de la pile*. Plus la *hauteur maximum* est grande, plus la *longueur de la pile* s'approche de la *longueur maximum théorique*. Cependant, cette hypothèse n'est pas valable dans le cas où les reads présents dans la pile correspondent à l'extrémité d'un transcrit. Dans ce cas précis, plus le $k-mer$ *de référence* est choisi près de cette extrémité, plus la *longueur de la pile* sera courte. La position du $k-mer$ *de référence* dans une pile de faible longueur donne **une indication sur l'extrémité du transcrit observé**. Si le $k-mer$ *de référence* est proche de la borne inférieure de la pile, alors il est très probable qu'une extrémité 3' soit observée. Dans cas inverse, il est très probable qu'une extrémité 5' soit observée. En conclusion, l'information sur la longueur de la pile est très intéressante à étudier dans le but de détecter de façon précise les bornes des transcrits afin de poser des points d'arrêt à la reconstruction des *meta-reads*. Il faut cependant veiller à contrôler le bruit de fond qui pourrait s'expliquer par des reads empilés qui racontent des histoires différentes des autres reads.

La *hauteur relative de la pile* apporte une information très utile afin de contrôler ce bruit de fond présent dans la pile comme nous allons le voir dans la partie suivante avec le *profil de la pile*.

3.2.5 Profil de pile, meta-support, et support vérité dans un empilement

Le *profil de la pile* (noté ρ) définit pour chaque position $q \in [inf(pile), sup(pile) - k]$ de la pile sa *hauteur relative* noté ρ_q.

Le *profil de la pile* permet d'accéder à la couverture en $k-mer$ de la pile le long de l'axe x. Le *profil de la pile* peut être modélisé par une fonction monotone croissante en dessous de 0, monotone décroissante au dessus de 0 et dont la dérivée s'annule en 0. La valeur maximum du profil de la pile sera donc toujours atteinte à la position 0 qui

correspond au *facteur de référence*.

Le *profil de la pile* peut être utilisé afin de définir les *bornes de confiance* de la pile dans l'objectif de limiter l'impact du bruit de fond à ses extrémités qui sont par nature moins couvertes que le centre de la pile. Une des techniques permettant de fixer ces nouvelles bornes est l'utilisation de la moyenne comme on peut l'observer dans la partie **C** de la figure 3.2. La moyenne fixe alors un seuil avec lequel on définit de nouvelles bornes $[x,y]$ avec $x < y, x \geq inf(pile), y \leq sup(pile)$.

Des méthodes plus sophistiquées permettraient de définir les *bornes "de confiance"* en détectant les points de rupture du *profil de la pile* et/ou en utilisant des notions de probabilité afin de définir une mesure de la confiance dans le signal observé à une position de la pile.

Le *profil de la pile* apporte également une information qualitative sur les transcrits de la même façon que la *longueur de la pile*. En effet, il serait possible d'observer des fins ou débuts de transcriptions alternatives qui seraient représentées par une décroissance discontinue du *profil de la pile* de part et d'autre de l'origine. Cela suppose cependant une modélisation fine du *profil de la pile* afin de détecter avec précision un événement de ce type.

Le *meta-support* définit pour chaque position $q \in [inf(pile), sup(pile) - k]$ de la pile une liste contenant le support des k-mers des reads qui chevauchent cette position. Pour obtenir le support du $k-mer$ d'un read i à une position q de la pile, on peut interroger son profil de support à la position $q - (p + inf(pile)$.

Le *meta-support* est une compilation des *profils de support* de reads qui ont été projetés sur l'axe x de la pile. Nous allons voir dans la suite comment le *meta-support* peut être utilisé afin d'obtenir des informations sur la pile et les différentes *histoires* qui la compose.

Le *support de vérité* définit pour chaque position q de la pile un rapport entre le nombre d'occurrences du $k-mer$ le plus représenté et la *hauteur relative* de la pile à cette position. Ce rapport, compris entre 0 et 1, est calculé à partir du *meta-support* et donne une information sur la proportion de reads racontant la même histoire à une position donnée de la *pile*.

Le *support de vérité* n'est pas directement calculé sur les séquences des $k-mers$ mais depuis leur support dans la collection de reads. Il est très peu probable qu'à une

position donnée de la pile, deux $k-mers$ différents aient la même valeur de support. Cette simplification permet de réduire la complexité de la procédure de calcul du support de vérité. En effet, la complexité de la requête Q_1 implémentée par les GkArrays est presque constante.

Une fois le *support de vérité* construit, il est possible de le questionner afin de savoir si, à une certaine position de l'empilement, l'ensemble des reads racontent la même histoire.

3.3 Algorithme de *RASTAck*

L'algorithme de *RASTAck* est une **implantation naïve des concepts** présentés précédemment avec pour objectif de reconstruire les *meta-reads*. Cette première version de l'algorithme est, d'une part une **preuve de concept** de notre nouvelle méthodologie d'assemblage, et d'autre part une base de réflexion. A partir de cette version, des premiers résultats pourront être générés et étudiés via des "métriques" afin de mesurer les gains en précision et sensibilité lors des améliorations successives qui vont être apportées à l'algorithme. Nous verrons dans le chapitre 4 comment les résultats de la méthode seront étudiés et comparés aux méthodologies concurrentes.

Le principe de l'algorithme de *RASTAck* est d'utiliser le concept de *piles* de reads afin d'obtenir **un assemblage local des meta-reads** et d'utiliser un processus itératif afin d'**étendre cet assemblage local jusqu'à la reconstruction complète d'un meta-read**. L'algorithme va donc passer en revue l'ensemble des reads de la collection dans le but de reconstituer toutes les *histoires* du transcriptome séquencé et marquer, lors de chaque assemblage local, les reads qui ont été utilisés.

Nous allons maintenant voir en détail les différentes étapes de l'algorithme et la manière dont ces dernières sont mises en place. Les étapes successives sont :

1. Choisir un "bon" read dans la collection
2. Choisir un "bon" $k-mer$ pour créer l'empilement
3. Définir les bornes de confiance de la pile
4. Choisir les $k-mers$ suivants pour la procédure d'extension
5. Marquer des reads utilisés
6. Terminer la reconstruction ou continuer à étendre

3.3.1 Choisir un “bon” read dans la collection

La première étape pour démarrer l'assemblage d'un *méta-read* est le **choix d'un read qui n'a pas encore été utilisé lors d'une précédente reconstruction**. Ce choix est facilité par la mise en place d'un vecteur de bits de taille N, le nombre de reads dans la collection (requête Q_a). Le choix d'un read non-utilisé n'est pas une condition suffisante pour débuter l'assemblage d'un *meta-read* car on souhaiterait s'assurer que le read en question raconte une seule *histoire*. Afin de s'en assurer il est possible de calculer un **“critère de beauté”**.

Nous avons décidé d'utiliser le “read-score” du *profil de support* et de fixer un seuil permettant de déterminer si **le score calculé est acceptable pour le choix d'un “bon” read**.

3.3.2 Choisir un bon $k-mer$ pour créer l'empilement

Une fois que le read initiateur est choisi, il est nécessaire de sélectionner un de ses $k-mers$ afin de créer la pile, ce qui nous permettra d'étendre la séquence du meta-read. Le choix du $k-mer$ est très important et doit respecter les critères suivants :

— le $k-mer$ ne contient pas d'erreur de séquençage.
Cela nous amènerait à créer un empilement vide ou presque.
— le $k-mer$ n'est pas répété de façon multiple dans le génome.
Cela nous amènerait à créer un empilement de reads ne racontant pas la même histoire
— le $k-mer$ ne contient pas un SNP.
Cela nous amènerait à créer un empilement avec une atténuation du signal du meta-read.

Comme pour le choix du read, le *profil de support* est une source d'information intéressante pour faire le choix du $k-mer$ d'empilement. Nous avons donc décidé de choisir le $k-mer$ possédant la valeur de support la plus proche de la valeur moyenne du profile de support. Les contraintes imposées sur le choix d'un “bon” $k-mer$ dans un “bon” read semblent suffisantes pour ne pas empiler des reads à partir d'*un mauvais* $k-mer$.

3.3.3 Définir les "bornes de confiance" de la pile

Il est maintenant possible de questionner la *collection de reads* à l'aide de la requête Q_2 pour établir l'empilement de reads.

L'étape qui suit la création de la pile est la recherche des *"bornes de confiance"* permettant de définir une zone de la pile où le signal du RNA-Seq est suffisamment grand pour réaliser un assemblage local. Comme nous l'avons vu précédemment, les *"bornes de confiance"* sont fixées à partir d'un seuil σ dépendant de la moyenne $\bar{\rho}$ du *profil de la pile* et d'un facteur τ de *tolérance à la moyenne* :

$$\sigma = \bar{\rho} \times \tau$$

La borne inférieure sera fixée à la plus petite position du *profil de la pile* ayant une valeur supérieure ou égale à σ. La borne supérieure sera fixée à la plus grande position du *profil de la pile* ayant une valeur supérieure ou égale à σ.

3.3.4 Choisir les k-mers suivants pour la procédure d'extension

Une fois les *"bornes de confiance"* définies, nous allons nous intéresser au *profil de vérité* afin de choisir le/les k-mer(s) qui vont servir à poursuivre la reconstruction ou à déterminer la fin de l'assemblage du meta-read.

Une manière simple de choisir le(s) $k-mer(s)$ d'extension est de se positionner aux extrémités de la pile définies par les "bornes de confiance" et de se rapprocher, pas à pas, vers l'origine de l'axe jusqu'à trouver un k-mer, de sorte que les deux conditions suivantes soient satisfaites :

1. La position de la pile où le $k-mer$ est choisi doit avoir un support de vérité supérieur à un seuil qui permettra de vérifier qu'une seule histoire se déroule à cette position de la pile. Le choix du seuil devrait être dynamique en fonction de la hauteur de la pile, afin de ne pas pénaliser une pile de faible hauteur et dont un des reads contiendrait une erreur de séquençage.
2. Le support du $k-mer$ choisi (requête Q_1 de la collection de reads) doit être cohérent avec le support du $k-mer$ *de référence*, afin d'une part de ne pas choisir un $k-mer$ qui soit répété dans le génome et d'autre part de s'assurer que l'histoire est toujours la même.

Si la recherche des $k-mers$ d'extension n'aboutit pas, cela veut dire que la reconstruction du méta-read est achevée.

3.3.5 Marquer les reads utilisés

Avant de procéder à une nouvelle itération de l'algorithme, nous allons devoir marquer les reads utilisés lors de l'extension afin de ne pas les piocher à nouveau lors d'un prochain empilement. Il est donc nécessaire de marquer les reads utilisés pour cet assemblage "local".

Cependant, comme nous l'avons vu précédemment, un read peut raconter plus d'une histoire. Afin de simplifier ce problème, nous avons décidé dans un premier temps de marquer les reads alignés dans la pile qui ont une position inférieure à celle des $k-mers$ de référence. Cela veut donc dire que leur séquence est complètement incluse dans l'assemblage local. Il reste tout de même une difficulté lorsque la reconstruction du méta-read est achevée car une grande partie des reads de la pile racontent une seconde histoire (dans le cas où l'empilement ne se trouve pas sur l'extrémité d'un transcrit).

3.3.6 Terminer la reconstruction ou continuer l'extension

Cette dernière étape est conditionnelle, si aucun $k-mer$ d'extension n'a été identifié alors la reconstruction est terminée, sinon il faut poursuivre l'assemblage jusqu'à la reconstruction complète du meta-read.

Dans le premier cas, on enregistre la séquence complète du meta-read sous forme d'une entrée d'un fichier FASTA de séquence. On peut placer en entête de la séquence les reads qui ont permis d'assembler le meta-read, afin de garder une trace de l'assemblage.

Dans le second cas, nous allons démarrer un processus récursif sur le(s) $k-mer(s)$ d'extension. En effet, nous allons recommencer le processus depuis l'étape 3, qui va définir un nouvel empilement de reads depuis ce(s) $k-mer(s)$, à la différence que, lors des récursions, nous allons nous intéresser uniquement à une seule des extrémités de la pile afin de ne pas effectuer de "retour arrière" dans la reconstruction.

Chapitre 4

Résultats

Sommaire

Nous allons voir dans ce chapitre les différents résultats qui ont été produits lors de ce stage. Dans une première partie nous allons nous intéresser aux outils que j'ai mis en place en début de stage afin de fournir un support à la mise au point de l'algorithme de RASTAck. Dans un deuxième temps nous discuterons de l'implémentation de l'algorithme de RASTAck dans le langage de programmation C++. Enfin nous nous intéresserons aux résultats préliminaires de cette première version de l'algorithme et la façon dont ces derniers seront analysés et comparés à d'autres approches concurrentes.

4.1 Outils de développement

Lors de mon stage, un premier travail a consisté à mettre en place des outils visant à faciliter l'utilisation d'une "collection de reads" et permettre la visualisation de cette collection. Deux outils ont été mis en place dans ce cadre. Le premier, nommé *GkDump*, est une version modifiée des GkArrays permettant la sauvegarde de la structure sur disque après sa construction afin de pouvoir la recharger plus tard. Le second outil, nommé *GkServer*, est une interface en ligne de commande permettant d'accéder aux reads de la collection à travers les différentes requêtes $Q_a, Q_b, Q_c, Q_1 \& Q_2$ définies dans la section 3.2.1 de ce rapport.

Ces deux outils ont permis de faciliter le développement de l'algorithme de *RASTAck* via une étude exploratoire des données.

4.1.1 GkDump : "Sauvegarder ses GkArrays sur disque"

Le premier travail de mon stage a consisté à implémenter un mécanisme de sérialisation pour la structure des GkArrays.

La sérialisation est un processus visant à coder l'état d'une information qui est en mémoire sous la forme d'une suite d'informations plus petites (dites atomiques) le plus souvent des octets, voire des bits. Cette suite pourra par exemple être utilisée pour la sauvegarde (persistance) ou le transport sur le réseau (proxy, RPC...). L'activité symétrique, visant à décoder cette suite pour créer une copie conforme de l'information d'origine, s'appelle la désérialisation (ou unmarshalling). [23]

Le mécanisme de sérialisation va nous permettre de **sauvegarder sous forme d'un fichier, binaire dans notre cas, l'ensemble des tables d'index des GkArrays** de manière linéaire. Il sera ensuite possible de "restaurer" la structure en mémoire depuis ce fichier binaire. L'objectif de la sérialisation dans notre contexte est de construire une seule fois la structure de données depuis une collection de reads et de pouvoir la réutiliser rapidement dans le cadre du développement de la méthode. En effet, il est évident que la mise au point d'un outil de traitement de données à large échelle est soumis à des étapes d'optimisation de la procédure. À chaque étape, il est essentiel de mesurer sur un même jeu de données les gains apportés par les améliorations de l'algorithme.

Il est également possible d'imaginer une seconde application de *GkDump* lorsque *RASTAck* sera mis en production : comme nous souhaitons intégrer les informations d'alignement des reads à notre méthode d'assemblage, il est nécessaire de passer par une étape de "mapping" des reads. Cette étape pourrait être prise en charge par CRAC[2], un des logiciels les plus précis et sensibles du marché, qui est dépendant des GkArrays pour son analyse des reads RNA-Seq. Il serait alors possible de construire une seule fois les tables des GkArrays depuis la collection de reads, et qui seraient utilisées successivement par CRAC puis RASTAck.

Afin de mettre ce mécanisme en place, j'ai utilisé une bibliothèque de développement C++ nommé `Boost::Serialization` qui propose une interface permettant de faciliter la sérialisation d'objet C++. J'ai alors ajouté pour chacune des classes de la librairie des

GkArrays une procédure décrivant comment les attributs d'une instance de la classe seront écrits sur un support linéaire et comment elles sont ensuite rechargées. C'est ensuite la bibliothèque `Boost::Serialization` qui utilise ces procédures et propose alors plusieurs supports de sauvegarde : fichier texte, document *XML* ou fichier binaire. Il est naturel dans notre cas de choisir le fichier binaire qui offre le meilleur taux de compression. Il y a néanmoins un désavantage à utiliser des fichiers binaires, c'est que ces derniers sont dépendants de l'architecture de la machine sur laquelle ils ont été générés. En effet, une sauvegarde des GkArrays effectuée sur une machine 32bits ne pourra être chargée sur une machine 64bits.

4.1.2 GkServer : "Questionner une collection de read"

Le deuxième outil que j'ai développé pendant ce stage, nommé "GkServer", est un logiciel utilisable en ligne de commande qui permet de charger une collection de reads, ici les GkArrays, de la questionner et d'afficher les résultats sous une forme graphique, facilitant ainsi l'accès à l'information.

L'objectif de cet outil est d'offrir un moyen d'accéder aux données des reads via les requêtes définies par une collection de reads afin de faciliter le développement d'un algorithme basé sur cette structure.

Une fois la collection de reads chargée en mémoire, plusieurs commandes sont disponibles :

- `info` : Récapitule un ensemble d'informations sur la **collection de reads**
  - Le nombre de reads présents dans la collection
  - La taille du $k-facteur$ qui a été utilisée pour l'indexation
  - La taille des reads si elle est fixe
  - Est-ce que les données sont paired-end ou single-end ?
  - Est-ce que les données sont orientées ?
- `help` : Propose un **message d'aide** avec une récapitulation des différentes commandes
- `tag <read_id>` : Récapitule différentes **informations sur un read** depuis son identifiant (voir figure 4.1) :
  - Sa *séquence*
  - Son *profil de support* sous forme graphique et numérique
  - La *moyenne* et *l'écart-type* du profil de support

— *L'alignement* du read s'il est disponible

— `stack <read_id> <pos>` : Créé **un empilement de reads** sur la base d'un $k-facteur$ de la collection et affiche l'empilement, son profil et ses différents supports (voir figure 4.2) :

— Les *propriétés* de la pile : longueur, hauteur

— Le *profil de la pile* (hauteur relative) sous forme graphique et numérique

— Le *support de vérité* sous forme graphique et numérique

— Les *bornes de confiances*

— *L'empilement de reads* : la séquence de chacun des reads est alignée sur *le facteur de référence*.

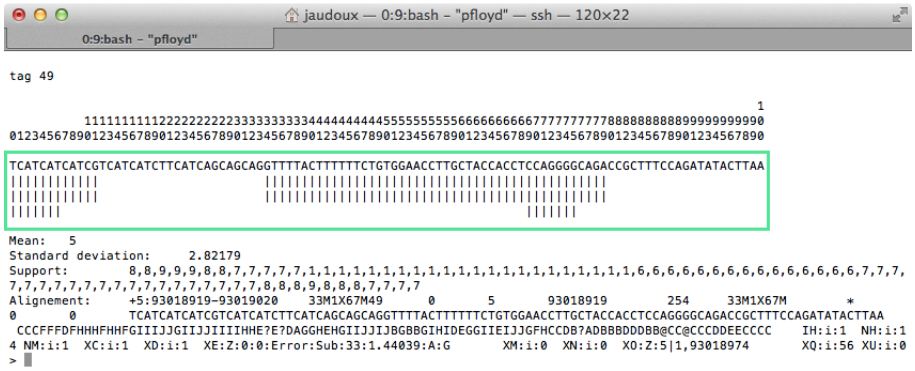

FIGURE 4.1 – Représentation d'un read (ou tag) de la collection par la commande `tag` de l'outil GkServer. L'encadré vert met en évidence le profil de support qui est projeté sur la séquence du read. Le profil de support est représenté à l'aide d'une échelle logarithmique ($log2$) afin d'observer les changements significatifs depuis une fenêtre réduite. On peut observer une baisse du support entre les positions 12 et 33 ce qui correspond à une erreur de séquençage. En dessous de l'encadré vert, on peut voir d'autres informations sur le read comme la moyenne et l'écart-type de son support, mais également son alignement sous la forme d'une entrée d'un fichier SAM.

L'utilisation de *GkServer* a permis de mettre en évidence de nombreuses notions développées dans le chapitre 3. Cet outil va également être utilisé pour étudier les résultats produits par l'implémentation de l'algorithme. En effet, si l'on observe des artefacts de reconstruction, il va être possible d'étudier les évènements qui se sont produits afin de proposer des solutions permettant d'améliorer l'algorithme.

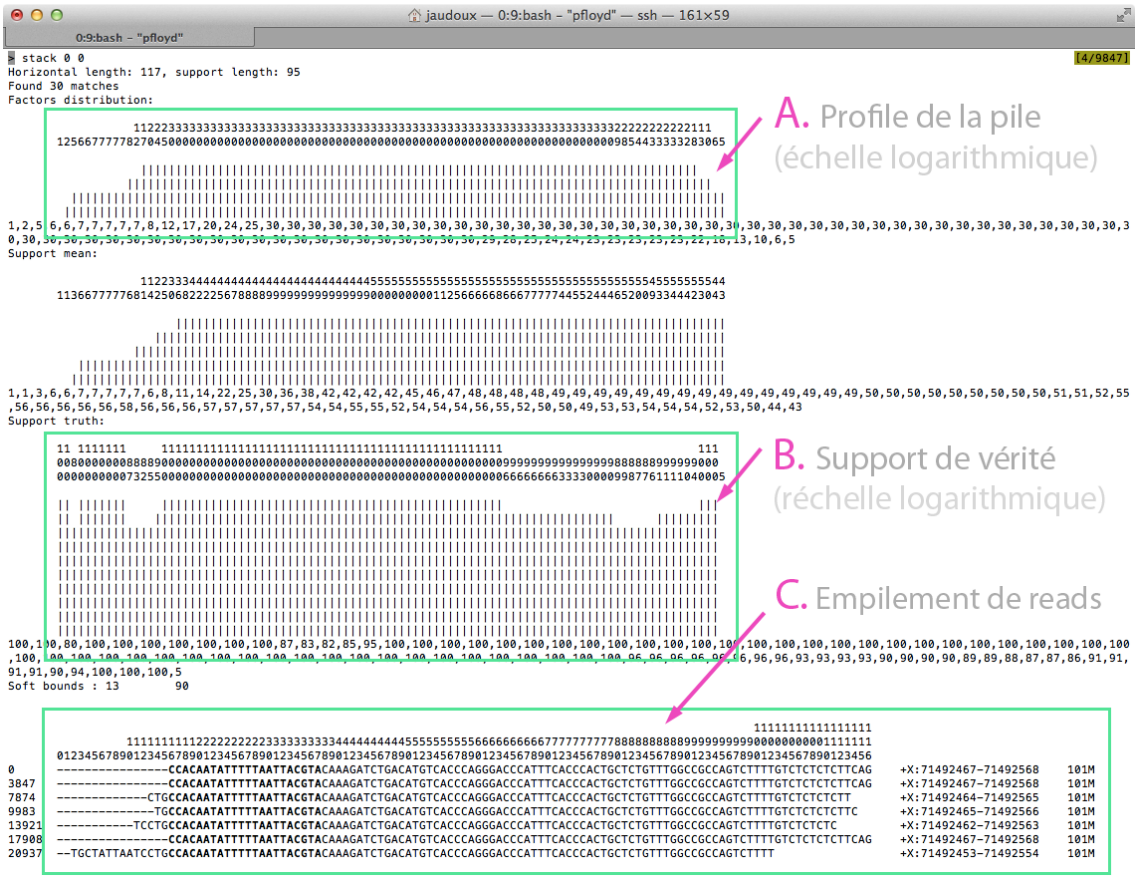

FIGURE 4.2 – Représentation d'en empilement de reads par commande `stack` de l'outil GkServer. L'encadré vert étiqueté *A* représente le profil de la pile représenté à l'aide d'une échelle logarithmique (log2). On peut observer que le profil n'est pas symétrique, les informations suivantes vont nous aider à comprendre pourquoi. Le second encadré vert étiqueté *B* représente le support de vérité. Ce dernier est relativement constant et ne passe jamais la barre des 0.9 ce qui permet de déduire qu'une seule histoire se déroule dans cette pile. Le dernier encadré vert étiqueté *C* représente la pile, elle-même. Les reads sont alignés sur le *facteur de référence* représenté ici en gras. Devant la séquence de chaque read est affiché leur identifiant dans la collection. Puis, après la séquence de chaque read est affiché l'alignement du read sur le génome de référence.

4.2 Implantation de *RASTack*

Nous allons voir dans cette section comment a été implémenté l'algorithme de *RASTAck* (présenté dans la section 3.3) sous différents aspects : quels ont été **les choix technologiques**, quelle est **l'architecture du code source** et quels ont été **les choix algorithmiques** réalisés pour l'implémentation de la méthode.

4.2.1 Langage et outils d'implémentation

L'implémentation de *RASTAck* a été réalisée dans le langage de programmation **C++**. D'une part cela permet un accès facile à *l'API* [1] *des GkArrays* qui est développée dans ce langage, d'autre part cela permet de gagner en performance grâce à un langage proche de la machine. Afin de faciliter la distribution du logiciel, une grande attention a été apportée au "packaging" en utilisant le projet *Autotools* de GNU [2]. Le projet Autotools permet d'établir un certain nombre de règles génériques de compilation et d'installation afin d'assurer la portabilité du logiciel sur les différents systèmes d'exploitation unix (ex : Linux, RedHat, MacOs, ...). Le code source de *RASTAck* est versionné sur git et hébergé sur la "forge" de l'INRIA [3] afin de faciliter les développements collaboratifs et le partage des sources.

4.2.2 Génie logiciel et architecture du code source

Nous allons présenter ici l'architecture du code source de *RASTAck* qui respecte les principes de la programmation orientée objet et dont l'objectif et de produire un code clair, compréhensible, facilement maintenable et évolutif.

L'architecture du code (voir figure 4.3) s'articule autour d'une classe principale, `RASTAck`, qui est le coeur du code et qui contient l'algorithme d'assemblage. Autour de cette classe, un ensemble d'outils permet de définir les différents concepts présentés dans le chapitre 3, dont notamment la `ReadCollection` qui implémente les 5 requêtes

1. En informatique, une interface de programmation (souvent désignée par le terme API pour Application Programming Interface) est un ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.
2. GNU est un système d'exploitation libre lancé en 1983 par Richard Stallman, puis maintenu par le projet GNU.
3. L'Institut national de recherche en informatique et en automatique (Inria) est un institut de recherche français en mathématiques et informatique.

permettant d'accéder à l'information des reads. La class `Read` implémente le concept de *profil de support* et permet le calcul des valeurs associées à ce support, moyenne, écart-type et "read-score". La classe `Stack` qui représente un empilement de `StackReadou` un `Read` est associée avec la position du *facteur de référence*. La classe `Bitvector` permet le marquage des `Reads` qui sont utilisés pendant la reconstruction. Enfin la classe `GenomeSearch` permet de générer un `Alignement` pour un `Read` donné.

L'architecture de *RASTAck* a été conçue avec comme objectif d'apporter un maximum de flexibilité pour les futurs développements. Dans ce but, un niveau d'abstraction a été apporté à chaque fragment du programme susceptible d'évoluer afin de facilement intégrer de nouveaux changements. C'est le cas de la collection de reads, une classe abstraite `ReadCollection` définit les différentes méthodes qu'une collection de read doit fournir. Une classe fille `GkReadCollection` définit ces méthodes abstraites en utilisant la structure de GkArrays. De la même façon, `GenomeSearch` propose de générer des `Alignements` de reads, qui sont en réalité extraits d'un fichier d'alignement SAM par `SAMGenomeSearch` et `SAMAlignement`. Ces deux niveaux d'abstraction permettent d'envisager l'utilisation d'une nouvelle structure pour répondre aux requêtes sur une collection de reads, ou d'utiliser une autre source d'un fichier SAM pour obtenir des alignements, comme par exemple utiliser une instance de *BLAT*[24] sur un serveur distant. génome*.

4.3 Analyse des résultas de *RASTAck*

Par contrainte de temps, les premiers résultats produits par l'implémentation de *RASTAck* n'ont subi qu'une analyse manuelle superficielle qui a permis de vérifier l'intérêt d'effectuer une analyse plus précise et chiffrée. Nous allons donc voir dans cette partie comment les résultats produits par l'implémentation de *RASTAck* vont pouvoir être analysés afin de permettre l'amélioration de la méthodologie d'assemblage et sa comparaison avec des méthodes concurrentes.

4.3.1 Données simulées et réelles

Afin d'étudier la sensibilité et la précision de notre méthode, il sera nécessaire d'utiliser des données simulées, dans le but de calibrer notre algorithme d'assemblage. L'utilisation de données simulées permettra l'optimisation de notre procédure d'assemblage

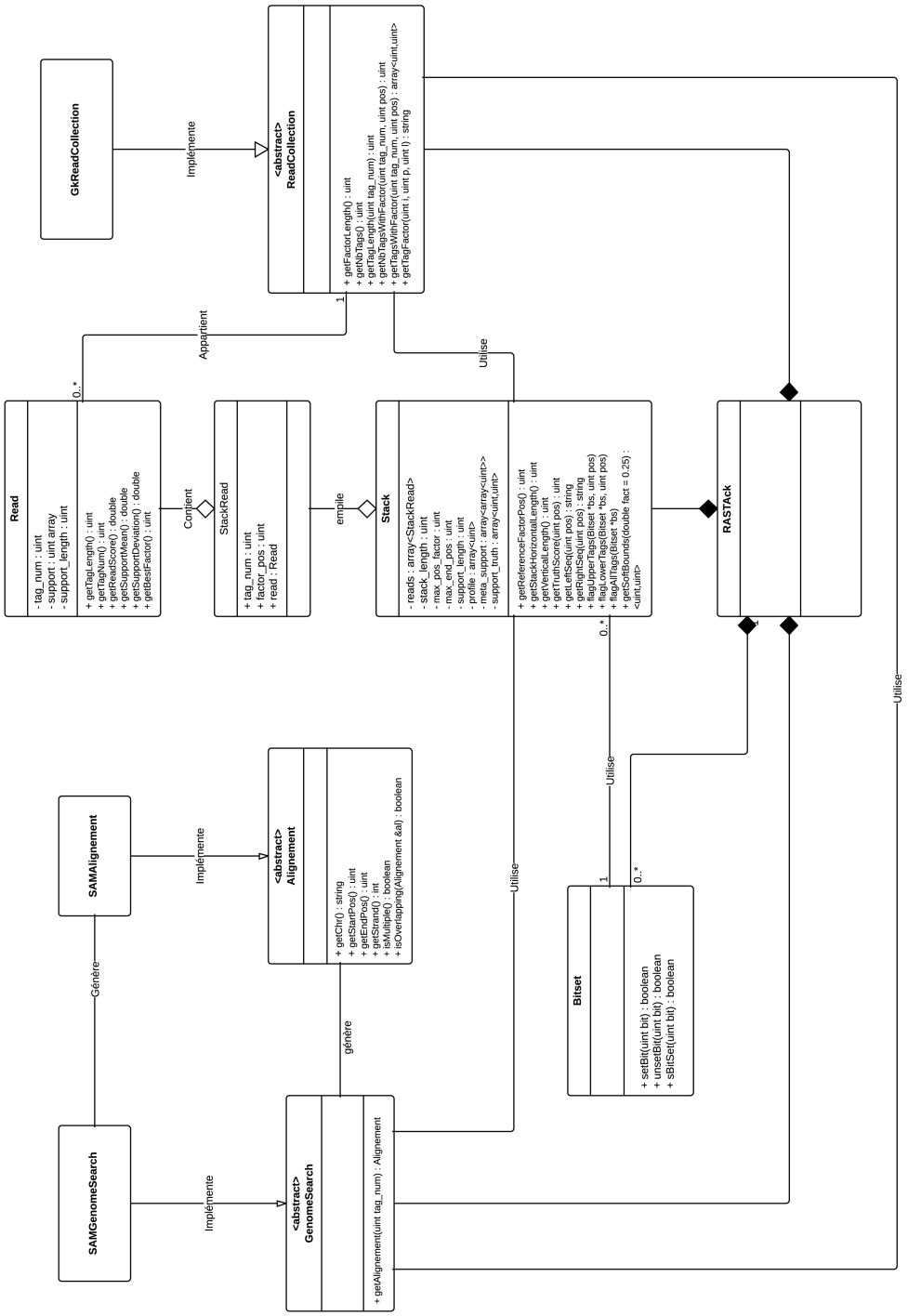

FIGURE 4.3 – **Diagramme de classes** Ce diagramme UML représente les différentes classes développées dans l'implémentation de *RASTAck* ainsi que les interactions entre elles.

puisque les meta-reads à assembler seront connus et permettront de servir de référence pour vérifier la pertinence de notre assemblage. Un tel jeu de données pourra être généré par le logiciel “FLUX simulator” [25], dont l’objectif est la génération d’une collection de reads depuis un transcriptome de référence.

Un second jeu de données réelles permettra de vérifier la fiabilité de notre approche en condition réelle.

4.3.2 Étude des paramètres

Lors des différentes analyses de notre méthode, il serait intéressant de faire varier différents paramètres. Un des paramètres essentiel de notre méthode d’assemblage est la taille, k, des facteurs indexés par la collection des reads. Il sera donc important de déterminer, en fonction de la donnée, quelle est la meilleure valeur à utiliser pour ce paramètre. Enfin, il serait intéressant de faire varier la nature de la donnée (longueur des reads et profondeur du séquençage) utilisée par *RASTAck* , afin d’identifier les besoins minimum de notre méthode qui permettent d’obtenir des résultats satisfaisants.

4.3.3 Étude de performances

Dans le cadre de l’analyse générale de notre méthode et non seulement des résultats produits, il serait intéressant de réaliser une étude sur les performances de l’implémentation de *RASTAck*. La mesure des performances comprend la consommation mémoire maximale requise par notre méthode, ainsi que le temps de calcul nécessaire avec un seul ou plusieurs processus (multithreading).

4.3.4 Étude des meta-reads

Des mesures spécifiques à la reconstruction des meta-reads pourraient être calculées depuis des données simulées, afin de mesurer notre capacité à reconstruire en intégralité l’ensemble des *meta-reads*. Ces mesures seraient à la fois quantitative : *“Quel est le pourcentage de meta-reads qui ont été reconstruits” ?*, et qualitative : *“Quel est le niveau de précision avec lequel nous détectons les bornes des meta-reads ?”*.

4.3.5 Comparaison à d'autres approches

Afin de comparer *RASTAck* aux logiciels concurrents, il sera nécessaire de mettre en place une méthode d'assemblage des meta-reads, afin de former les transcrits complets. Il sera alors possible d'utiliser les mesures définies dans l'étude comparative de référence [26], afin de déterminer sur quels critères les méthodes concurrentes seront comparées les unes aux autres.

Chapitre 5

Discussion

Dans ce dernier chapitre, nous allons discuter de la méthode qui a été proposée au chapitre 3 et implantée dans le chapitre 4. Nous allons tout d'abord parler des points forts de *RASTAck* ainsi que des difficultés restantes. Enfin, nous discuterons des perspectives de ce stage et notamment des possibilités d'amélioration de l'algorithme de *RASTAck*.

5.1 Méthode proposée

La méthode qui a été proposée dans ce rapport représente la première tentative d'assemblage d'une collection de reads indexée dans une structure de données sur l'ensemble de ces $k-mers$. Il existe plusieurs avantages à développer cette approche. D'une part, l'information complète de la séquence des reads est mise à contribution pour l'assemblage : il n'y a pas un échantillonnage du signal comme c'est le cas dans une approche par graphe de de Bruijn. D'autre part, la méthode que nous proposons n'est pas directement dépendante d'un génome de référence car l'assemblage se fait sur un critère de ressemblance entre les séquences. Toutefois, en conservant l'information complète sur les reads, il est possible de développer une approche intégrée visant à combiner plusieurs sources d'information pendant l'assemblage des reads, comme leurs alignements.

Cette nouvelle approche, que nous avons appelée *RASTAck*, est permise grâce à une structure de données (les GkArrays) qui indexe les $k-mers$ des reads et qui permet de répondre à un ensemble de cinq requêtes décrites dans la section 3.2.1 de ce rapport. Cette structure est actuellement le seule capable de répondre aux requêtes nécessaires à notre algorithme d'assemblage. L'avantage des GkArrays est que les indexs de la struc-

ture sont chargés en mémoire, ce qui permet d'obtenir un temps d'exécution des requêtes très performant et donc permet de réaliser l'assemblage des reads en peu de temps. De plus, notre méthode d'assemblage s'intéresse aux reads par pile et non de façon unitaire. Cette approche par pile permet de réduire la complexité de l'assemblage car il est possible de vérifier à un point de la pile que l'ensemble des reads raconte la même histoire. **Une fois la structure d'indexation construite, moins de 2 minutes sont alors nécessaires pour assembler les meta-reads depuis une collection de 12 millions de reads (75pb) RNA-Seq de données simulées à l'aide du transcriptome de la drosophile, et ce avec un seul thread.** Le revers de cette rapidité d'exécution est l'impact mémoire des GkArrays, qui pour ce même jeu de 12 millions de reads, s'élève à 6 Gigaoctets. Récemment, un ensemble d'optimisations a été proposé, afin de réduire l'impact mémoire des GkArrays [27]. Cependant, l'implantation de la structure proposée par cette étude n'est pas disponible.

L'approche actuelle de notre méthodologie d'assemblage est la reconstruction des *meta-reads* qui sont une unité de reconstruction inférieure au transcrit qui constitue le but final d'un algorithme d'assemblage du RNA-Seq. La reconstruction des meta-reads possède des avantages car ce sont des éléments atomiques du transcriptome qui, une fois assemblés correctement les uns avec les autres, permettent de former l'ensemble de transcrits. Par ailleurs, les *meta-reads* peuvent servir d'unité de comptage afin de réaliser une analyse différentielle entre plusieurs jeux de données RNA-Seq.

Enfin, ce stage a permis le développement d'une nouvelle méthode d'assemblage du RNA-Seq et de son implantation dans le langage C++. Des premiers résultats ont pu être générés depuis un jeu de données simulées et une première analyse superficielle des données a permis de constater que la méthodologie génère très peu d'artefacts et que les reconstructions des meta-reads proposées semblent encourageantes. Une étude approfondie, à l'aide des métriques, proposée dans la section 4.3 permettra de vérifier la qualité de la reconstruction et de proposer des améliorations à la méthode.

5.2 Perspectives

Nous allons voir dans cette section les nombreuses perspectives de ce stage.

5.2.1 Affiner les étapes de l'algorithme de RASTAck

La première perspective de ce stage est de continuer à améliorer la méthode proposée dans le chapitre 3 à chacune des étapes. Un des points de départ des améliorations sera l'analyse des résultats. Mais, certaines améliorations théoriques sont également envisageables comme nous allons le voir prochainement.

La première étape qui consiste à *choisir un bon read dans la collection* pourrait être améliorée en réalisant une analyse de la capacité du "read-score" à prédire la présence de plusieurs histoires dans un read. On pourrait par exemple utiliser une méthode d'apprentissage (type SVM) sur des données simulées, afin de déterminer une seuil partageant les reads selon leur "read-score" et permettant de regrouper d'un côté, ceux racontant une seule histoire, et de l'autre, ceux racontant plusieurs histoires.

La seconde partie qui consiste à *choisir un "bon" $k-mer$ dans le read* pourrait être améliorée à l'aide d'une analyse fine du profil de support dans le but de déterminer le $k-mer$ le plus représentatif de l'histoire que l'on cherche à reconstruire.

La troisième partie qui consiste à *définir les bornes de confiance* repose pour l'instant sur un écart par rapport à la moyenne du profil de la pile. Le choix des bornes pourrait être amélioré par la mise en place d'un modèle probabiliste permettant de définir le seuil de couverture à une position de la pile nécessaire pour déterminer que l'ensemble des reads racontent la même histoire. De plus, une étude du profil de la pile permettrait de mettre en évidence des débuts et fins de transcription alternative qui sont des événements pris en compte par le modèle du "meta-read" et non dans notre méthode actuelle d'assemblage.

La quatrième partie qui consiste à *choisir les $k-mers$ suivants pour le processus d'extension* pourrait être optimisée afin d'effectuer des "sauts" dans la pile, à la manière d'une recherche dichotomique, afin de trouver plus rapidement les candidats d'extension.

Enfin, la cinquième étape consistant à marquer les reads utilisés ne propose pour l'heure que le marquage complet d'un read. Néanmoins, nous avons vu qu'un même read peut être impliqué dans deux histoires différentes. Il serait donc intéressant d'identifier une structure permettant de marquer uniquement la partie du read qui a été impliquée dans la reconstruction d'un *meta-read*, afin de pouvoir piocher ce même read à nouveau.

5.2.2 Intégrer les informations d'alignement et plus encore

Une seconde perspective de ce stage est le développement d'une approche intégrée que nous avons mis en avant dans la description de la "philosophie de *RASTAck*" (voir chapitre 3).

Dans le cadre de cette perspective, il serait intéressant d'intégrer l'information sur l'alignement des reads dans l'algorithme d'assemblage de *RASTack*. Comme nous l'avons vu dans le chapitre 4, l'architecture de RASTAck comporte déjà les "briques" logicielles nécessaires pour faire le lien entre un read de la collection et son alignement. Ce dernier pourrait alors s'intégrer facilement dans les étapes de notre méthode.

Par exemple, l'alignement pourrait servir, lors de la deuxième étape, au choix du bon k-mer, afin de sélectionner une portion du read qui ne contient pas de variant polymorphique. L'alignement permettrait également de définir avec plus de précision les bornes des exons et donc les bornes des meta-reads qui impliquent un événement d'épissage alternatif. On pourrait en outre identifier les reads qui ont été empilés sur la base d'un facteur répété dans le génome et qui par définition ne racontent pas la même histoire.

Une seconde information qui pourrait être facilement intégrée à notre processus de reconstruction est le paired-end qui est déjà pris en compte par les GkArrays. Il serait possible d'utiliser cette information pour lever des ambigüités lors de l'assemblage de séquences répétées dans le génome.

5.2.3 Une structure d'indexation plus satisfaisante

Une dernière perspective de ce stage est la mise au point ou l'identification d'une nouvelle structure de données permettant de répondre aux requêtes définies par la *Collection de reads*. En effet, la structure des GkArrays a un impact mémoire très important qui croît avec la taille de la collection de reads indexée. Des approches de structures succinctes [28] ont été proposées pour la représentation des graphes de de Bruijn. Sur le même modèle, on pourrait également proposer une structure succincte permettant d'indexer une collection de reads sur leur $k-mers$.

Conclusion

Ce rapport constitue l'aboutissement de mon stage de Master 2 effectué au sein de l'équipe de Thérèse Commes à l'Institut de Recherche en Biothérapie. Pendant ce stage, je me suis intéressé au traitement des données issues des séquenceurs haut débit, dans le cadre de l'étude des transcriptomes. Ces données constituent un immense puzzle du transcriptome qui a été séquencé. L'objectif des chercheurs en bioinformatique est alors de proposer des structures et des algorithmes permettant d'assembler ces données, afin de former les transcrits initialement présents dans la cellule. Ce problème d'assemblage est cependant très complexe et aucune des méthodes actuellement présentées n'est capable de fournir un assemblage de qualité.

Après une présentation du contexte biologique et technologique, nous nous sommes intéressé à l'état de l'art des méthodes et des structures proposées pour reconstruire des transcrits. Nous avons alors identifié une structure de données, les GkArrays, qui n'a jamais été utilisée dans le cadre de la reconstruction des transcrits et qui présente la capacité d'indexer une collection de reads sur leur $k-mer$ permettant ainsi d'empiler un ensemble de reads qui partagent une sous-séquence commune. Nous avons alors proposé un ensemble de concepts ainsi qu'un algorithme, afin de mettre au point une nouvelle méthode, nommée *RASTAck*, permettant d'assembler les reads RNA-Seq grâce au GkArrays en formant des piles successives de reads.

Cette nouvelle méthode, à l'inverse des méthodes "de novo" actuelles, utilise l'ensemble de la séquence des reads, afin d'utiliser au mieux le signal produit par le RNA-Seq. Notre méthode se place donc à l'échelle du read permettant ainsi le développement d'approches intégrées permettant, par exemple, d'utiliser l'information sur l'alignement des reads dans le processus de reconstruction. Aucune des méthodes actuelles n'est capable de faire ce rapprochement entre approche "guidée par génome" et approche "de novo".

Nous avons ensuite étudié les résultats de ce stage et notamment comment *RASTAck* a été implanté en langage C++, afin de vérifier l'intérêt de l'algorithme que nous pro-

posons. Les résultats préliminaires sont encourageants et nous amènent à réfléchir à la mise en place de mesures permettant d'optimiser les différentes étapes de reconstruction. Nous avons présenté un certain nombre d'améliorations de notre méthode dans la section "Perspectives" du chapitre "Discussion".

Afin de conclure, ce stage de Master 2 m'a permis de vérifier mon affection pour la bioinformatique et ma volonté de réaliser une thèse dans ce domaine qui débutera en septembre prochain. Ce stage m'a amené à réfléchir sur l'élaboration d'une nouvelle méthode de traitement de séquences génétiques depuis la recherche bibliographique jusqu'à l'implémentation logicielle et la rédaction de ce document. Celui-ci représente le travail de recherche sur lequel j'ai été amené à travailler en intégralité et je remercie Nicolas Philippe de m'en avoir donné l'opportunité.

Bibliographie

[1] N. Philippe, M. Salson, T. Lecroq, M. Léonard, T. Commes, and E. Rivals, "Querying large read collections in main memory : a versatile data structure," *BMC Bioinformatics*, vol. 12, p. 242, June 2011.

[2] N. Philippe, M. Salson, T. Commes, and E. Rivals, "CRAC an integrated approach to the analysis of RNA-seq reads," *Genome biology*, vol. 14, p. R30, Mar. 2013.

[3] N. Philippe, E. Bou Samra, A. Boureux, A. Mancheron, F. Rufflé, Q. Bai, J. De Vos, E. Rivals, and T. Commes, "Combining DGE and RNA-sequencing data to identify new polyA+ non-coding transcripts in the human genome," *Nucleic Acids Research*, vol. 42, pp. 2820–2832, Mar. 2014.

[4] Wikipedia, "Alternative splicing," July 2014. Page Version ID : 614914242.

[5] B. J. Haas and M. C. Zody, "Advancing RNA-seq analysis," *Nature Biotechnology*, vol. 28, pp. 421–423, May 2010.

[6] C. Trapnell, B. A. Williams, G. Pertea, A. Mortazavi, G. Kwan, M. J. van Baren, S. L. Salzberg, B. J. Wold, and L. Pachter, "Transcript assembly and quantification by RNA-seq reveals unannotated transcripts and isoform switching during cell differentiation," *Nature biotechnology*, vol. 28, pp. 511–515, May 2010.

[7] M. Guttman, M. Garber, J. Z. Levin, J. Donaghey, J. Robinson, X. Adiconis, L. Fan, M. J. Koziol, A. Gnirke, C. Nusbaum, J. L. Rinn, E. S. Lander, and A. Regev, "Ab initio reconstruction of cell type-specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincRNAs," *Nature Biotechnology*, vol. 28, pp. 503–510, May 2010.

[8] L. Song and L. Florea, "CLASS : constrained transcript assembly of RNA-seq reads," *BMC Bioinformatics*, vol. 14, p. S14, Apr. 2013.

[9] N. Boley, M. H. Stoiber, B. W. Booth, K. H. Wan, R. A. Hoskins, P. J. Bickel, S. E. Celniker, and J. B. Brown, "Genome-guided transcript assembly by integrative

analysis of RNA sequence data," *Nature Biotechnology*, vol. 32, no. 4, pp. 341–346, 2014.

[10] M. G. Grabherr, B. J. Haas, M. Yassour, J. Z. Levin, D. A. Thompson, I. Amit, X. Adiconis, L. Fan, R. Raychowdhury, Q. Zeng, Z. Chen, E. Mauceli, N. Hacohen, A. Gnirke, N. Rhind, F. di Palma, B. W. Birren, C. Nusbaum, K. Lindblad-Toh, N. Friedman, and A. Regev, "Full-length transcriptome assembly from RNA-seq data without a reference genome," *Nature Biotechnology*, vol. 29, pp. 644–652, July 2011.

[11] Y. Xie, G. Wu, J. Tang, R. Luo, J. Patterson, S. Liu, W. Huang, G. He, S. Gu, S. Li, X. Zhou, T.-W. Lam, Y. Li, X. Xu, G. K.-S. Wong, and J. Wang, "SOAPdenovo-trans : de novo transcriptome assembly with short RNA-seq reads," *Bioinformatics (Oxford, England)*, vol. 30, pp. 1660–1666, June 2014.

[12] G. Robertson, J. Schein, R. Chiu, R. Corbett, M. Field, S. D. Jackman, K. Mungall, S. Lee, H. M. Okada, J. Q. Qian, M. Griffith, A. Raymond, N. Thiessen, T. Cezard, Y. S. Butterfield, R. Newsome, S. K. Chan, R. She, R. Varhol, B. Kamoh, A.-L. Prabhu, A. Tam, Y. Zhao, R. A. Moore, M. Hirst, M. A. Marra, S. J. M. Jones, P. A. Hoodless, and I. Birol, "De novo assembly and analysis of RNA-seq data," *Nature Methods*, vol. 7, pp. 909–912, Nov. 2010.

[13] P. Jain, N. M. Krishnan, and B. Panda, "Augmenting transcriptome assembly by combining de novo and genome-guided tools," *PeerJ*, vol. 1, p. e133, 2013.

[14] "Augmenting transcriptome assembly by combining de novo... [PeerJ. 2013] - PubMed - NCBI."

[15] J. T. Simpson and R. Durbin, "Efficient construction of an assembly string graph using the FM-index," *Bioinformatics*, vol. 26, pp. i367–i373, June 2010.

[16] N. Philippe, A. Boureux, L. Bréhélin, J. Tarhio, T. Commes, and E. Rivals, "Using reads to annotate the genome : influence of length, background distribution, and sequence errors on prediction capacity," *Nucleic Acids Research*, vol. 37, pp. e104–e104, Aug. 2009.

[17] R. Chikhi and P. Medvedev, "Informed and automated k-mer size selection for genome assembly," *Bioinformatics (Oxford, England)*, vol. 30, pp. 31–37, Jan. 2014.

[18] Wikipedia, "Graphe de de bruijn," Jan. 2014. Page Version ID : 90865297.

[19] P. A. Pevzner, H. Tang, and M. S. Waterman, "An eulerian path approach to DNA fragment assembly," *Proceedings of the National Academy of Sciences*, vol. 98, pp. 9748–9753, Aug. 2001.

[20] S. Anders, A. Reyes, and W. Huber, "Detecting differential usage of exons from RNA-seq data," *Genome Research*, vol. 22, pp. 2008–2017, Oct. 2012.

[21] R. Patro, S. M. Mount, and C. Kingsford, "Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms," *Nature Biotechnology*, vol. 32, pp. 462–464, May 2014.

[22] C. Husquin, "Reconstituer le puzzle : depuis des fragments jusqu' 'a l'ARN," rapport de stage de master 1, LifL, Lille, June 2013.

[23] Wikipedia, "Sérialisation," June 2014. Page Version ID : 104915732.

[24] W. J. Kent, "BLAT–the BLAST-like alignment tool," *Genome Research*, vol. 12, pp. 656–664, Apr. 2002.

[25] T. Griebel, B. Zacher, P. Ribeca, E. Raineri, V. Lacroix, R. Guigó, and M. Sammeth, "Modelling and simulating generic RNA-seq experiments with the flux simulator," *Nucleic Acids Research*, vol. 40, pp. 10073–10083, Nov. 2012.

[26] T. Steijger, J. F. Abril, P. G. Engström, F. Kokocinski, The RGASP Consortium, T. J. Hubbard, R. Guigó, J. Harrow, and P. Bertone, "Assessment of transcript reconstruction methods for RNA-seq," *Nature Methods*, vol. 10, pp. 1177–1184, Dec. 2013.

[27] N. Välimäki and E. Rivals, "Scalable and versatile k-mer indexing for high-throughput sequencing data.," in *Proc. 9th International Symposium on Bioinformatics Research and Applications (ISBRA'13)*, pp. 237–248, May 2013.

[28] A. Bowe, T. Onodera, K. Sadakane, and T. Shibuya, "Succinct de bruijn graphs," in *Algorithms in Bioinformatics* (B. Raphael and J. Tang, eds.), no. 7534 in Lecture Notes in Computer Science, pp. 225–235, Springer Berlin Heidelberg, Jan. 2012.

ac mauris quis placerat. Vestibulum tempor lacus eros, vitae venenatis ipsum auctor ut. Nam risus justo, vulputate id lobortis ut, blandit eget nibh. Ut sed sapien et urna rhoncus placerat. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae ; Curabitur mattis id justo quis auctor. Mauris vel nunc nec urna posuere interdum. Interdum et malesuada fames ac ante ipsum primis in faucibus. Nulla pretium tempor purus eget fringilla. Nullam viverra ac leo non interdum. Vestibulum fringilla non nunc vitae interdum. Integer vel ligula vel ipsum commodo tristique vitae in neque. Sed euismod libero in diam eleifend, mollis posuere eros fringilla. Duis vehicula, erat eget mollis auctor, nibh diam consectetur tellus, in porta tortor neque in arcu. In ut velit a lacus adipiscing aliquet. Cras ultrices odio erat, lobortis auctor leo aliquet ut. ac mauris quis placerat. Vestibulum tempor lacus eros, vitae venenatis ipsum auctor ut. Nam risus justo,

vulputate id lobortis ut, blandit eget nibh. Ut sed sapien et urna rhoncus placerat. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae ; Curabitur mattis id justo quis auctor. Mauris vel nunc nec urna posuere interdum. Interdum et malesuada fames ac ante ipsum primis in faucibus. Nulla pretium tempor purus eget fringilla. Nullam viverra ac leo non interdum. Vestibulum fringilla non nunc vitae interdum. Integer vel ligula vel ipsum commodo tristique vitae in neque. Sed euismod libero in diam eleifend, mollis posuere eros fringilla. Duis vehicula, erat eget mollis auctor, nibh diam consectetur tellus, in porta tortor neque in arcu. In ut velit a lacus adipiscing aliquet. Cras ultrices odio erat, lobortis auctor leo aliquet ut.