# Assemblage adaptatif de génomes et de méta-génomes par passage de messages

**Thèse**

**Sébastien Boisvert**

**Doctorat en physiologie-endocrinologie**
Philosophiæ doctor (Ph.D.)

Québec, Canada

# Résumé

De manière générale, les procédés et processus produisent maintenant plus de données qu'un humain peut en assimiler. Les grosses données (*Big Data*), lorsque bien analysées, augmentent la compréhension des processus qui sont opérationnels à l'intérieur de systèmes et, en conséquence, encouragent leur amélioration. Analyser les séquences de l'acide désoxyribonucléique (ADN) permet de mieux comprendre les êtres vivants, en exploitant par exemple la biologie des systèmes. Les séquenceurs d'ADN à haut débit sont des instruments massivement parallèles et produisent beaucoup de données. Les infrastructures informatiques, comme les superordinateurs ou l'informatique infonuagique, sont aussi massivement parallèles de par leur nature distribuée. Par contre, les ordinateurs ne comprennent ni le français, ni l'anglais – il faut les programmer. Les systèmes logiciels pour analyser les données génomiques avec des superordinateurs doivent être aussi massivement parallèles. L'interface de passage de messages permet de créer de tels logiciels et une conception granulaire permet d'entrelacer la communication et le calcul à l'intérieur des processus d'un système de calcul. De tels systèmes produisent des résultats rapidement à partir de données. Ici, les logiciels RayPlatform, Ray (incluant les flux de travail appelé *Ray Meta* et *Ray Communities*) et Ray Cloud Browser sont présentés. L'application principale de cette famille de produits est l'assemblage et le profilage adaptatifs de génomes par passage de messages.

# Abstract

Generally speaking, current processes – industrial, for direct-to-consumers, or research-related – yield far more data than humans can manage. Big Data is a trend of its own and concerns itself with the betterment of humankind through better understanding of processes and systems. To achieve that end, the mean is to leverage massive amounts of big data in order to better comprehend what they contain, mean, and imply. DNA sequencing is such a process and contributes to the discovery of knowledge in genetics and other fields. DNA sequencing instruments are parallel objects and output unprecedented volumes of data. Computer infrastructures, cloud and other means of computation open the door to the analysis of data stated above. However, they need to be programmed for they are not acquainted with natural languages. Massively parallel software must match the parallelism of supercomputers and other distributed computing systems before attempting to decipher big data. Message passing – and the message passing interface – allows one to create such tools, and a granular design of blueprints consolidate production of results. Herein, a line of products that includes RayPlatform, Ray (which includes workflows called Ray Meta and Ray Communities for metagenomics) and Ray Cloud Browser are presented. Its main application is scalable (adaptive) assembly and profiling of genomes using message passing.

# Table des matières

# Liste des tableaux

# Liste des figures

I will not give you counsel, saying do this, or do that. For not in doing or contriving, nor in choosing between this course and another, can I avail ; but only in knowing what was and is, and in part also what shall be.

---

Galadriel

# Remerciements

Je remercie en premier ma conjointe Élénie avec laquelle j'ai toujours du plaisir à discuter d'idées et à m'amuser.

Je remercie ma mère Jocelyne et mon père Roger pour le support continu, ainsi que mon frère Maxime, qui est aussi un passionné d'informatique et de programmation de systèmes complexes.

Permettez-moi de souligner que faire un doctorat est un effort soutenu sur plusieurs années. Compléter un doctorat de recherche est pour moi un exploit remarquable dans un domaine d'excellence. Pendant ces nombreuses années que j'ai investi dans la recherche scientifique, plusieurs personnes m'ont aidé.

Je remercie Jacques Corbeil – professeur titulaire à la Faculté de médecine de l'Université Laval – qui a été mon directeur de recherche au doctorat et à la maîtrise. Il a été aussi le superviseur de deux de mes stages. Je le remercie pour tout le temps qu'il a investi dans ma formation et pour son support, sa très grande disponibilité et sa versatilité. Il est professeur au département de médecine moléculaire de l'Université Laval. Il est aussi un chercheur remarquable pour qui la multidisciplinarité n'a aucun secret et pour qui tout est possible. Il m'a donné énormément de liberté académique pendant mon doctorat, et cette liberté m'a permis d'accomplir des choses remarquables au sein de son groupe de recherche. De plus, sachant très bien l'importance du développement des compétences de communication dans le monde scientifique, il m'a prêté – au tout début de mon doctorat – le livre *The Elements of Style* par William Strunk, Jr. Cet ouvrage de référence contient des règles d'écriture pour la langue anglaise. Récemment, j'ai transmis ce livre à Charles Joly Beauparlant – un étudiant au doctorat codirigé par Jacques Corbeil.

Je remercie François Laviolette – mon codirecteur de mes travaux de doctorat – pour ses apports combinatoires à mes projets de recherche. Particulièrement, il est toujours

ouvert pour discuter de graphes, de la théorie des graphes, de la combinatoire et des liens qu'ont ses théories avec la biologie. Il est professeur au département d'informatique et de génie logiciel de l'Université Laval.

Je remercie aussi Mario Marchand, qui a été mon codirecteur de recherche lors de ma maîtrise à l'Université Laval, pour laquelle Jacques Corbeil a assumé la direction. Mario Marchand est une personne très pédagogique qui sait expliquer l'inexplicable.

Je remercie les trois examinateurs de mon examen de doctorat (4 avril 2011) pour m'avoir évaluer : Jacques Corbeil, François Laviolette et Arnaud Droit.

Je remercie les cinq examinateurs de mon séminaire de doctorat (20 décembre 2011) pour avoir évaluer ma recherche : Jean-François Bilodeau, Jacques Corbeil, François Laviolette, Christian Landry et Mario Marchand.

Je tiens à remercie les cinq examinateurs qui ont accepter de lire ma thèse et de la commenter : Jacques Corbeil, François Laviolette, André Darveau, Guillaume Bourque (Université McGill) et Sylvain Moineau.

Je remercie également Francine Durocher, la directrice de mon programme d'études. Elle m'a introduit au monde de la recherche lors d'un stage dans son laboratoire en 2005.

Je remercie mes collègues Frédéric Raymond, René Paradis, Charles Joly Beauparlant, Pier-Luc Plante, Maxime Déraspe, Jean-François Erdelyi, Thibault Varin, Frédéric Fournier, Arnaud Droit, et Francis Brochu pour les nombreuses discussions sur l'informatique, la bio-informatique, la recherche, et l'administration de systèmes.

Je suis particulièrement appréciatif de Frédéric Raymond, avec qui j'ai réalisé beaucoup de projets formidables. Il a été mon superviseur de stage lorsque j'ai commencé dans le domaine de la bio-informatique, en 2006. Il a été pour moi un mentor lorsque j'ai débuté dans le milieu de la recherche scientifique en bio-informatique. Je remercie aussi Lynda Robitaille qui est toujours disponible pour aider avec les tâches administratives.

Je remercie Angana Mukherjee, Adriano C. Coelho, Marc Ouellette, Barbara Papadopoulou, Philippe Leprohon pour les nombreuses collaborations portant sur le parasite *Leishmania*.

Je remercie les payeurs de taxes canadiens pour leur contribution financière à la recherche scientifique. En particulier, je remercie toutes les Canadiennes et tous les Ca-

nadiens pour avoir contribué financièrement à mes recherches au travers des Instituts de recherche en santé du Canada, plus précisément l'Institut de génétique. En effet, j'ai reçu une bourse Frederick Banting et Charles Best pour ma maîtrise ainsi qu'une bourse Frederick Banting et Charles Best pour mon doctorat.

Je remercie aussi la communauté scientifique. Je remercie Rick Stevens, Pavan Balaji et Fangfang Xia du laboratoire national américain Argonne (Argonne National Laboratory) pour m'avoir invité à présenter mes travaux à plusieurs reprises.

Je remercie également Daniel Gruner et Chris Loken pour leur intérêt dans mes travaux de doctorat. Daniel Gruner et Chris Loken sont à SciNet, à l'Université de Toronto. SciNet est une organisation qui opère des superordinateurs pour Calcul Canada. J'ai eu accès au seul IBM Blue Gene/Q au Canada, lequel est opéré par SciNet, pour y faire des tests divers.

Je remercie Nicholas J. Loman pour m'avoir invité pour que je présente mes travaux de thèse à la rencontre de bioinformatique *Beatles and Bioinformatics* le 27 novembre 2013.

Finalement, je tiens à remercier quatre personnes qui ont fait leur doctorat sur l'assemblage *de novo* pour des discussions enrichissantes : Rayan Chikki de l'École normale supérieure de Cachan – Antenne de Bretagne [53], Daniel Zerbino de l'Université de Cambridge [306], Jared Simpson de l'Université de Cambridge [274] et Mark Chaisson de l'Université de Californie à San Diego [45].

# Avant-propos

Les nouvelles technologies de séquençage de l'acide désoxyribonucléique analysent des milliards de molécules en parallèles, permettant désormais l'obtention de très volumineux ensembles de données biologiques. De telles quantités de séquences promettent de répondre à d'importantes questions scientifiques avec des implications en médecine, en énergie et pour l'économie. Cependant, il y a un besoin pressant pour des méthodologies parallèles pour transformer les séquences en connaissances.

Ce document est une thèse doctorat par articles. Il inclut mes résultats de recherche au doctorat de janvier 2010 à décembre 2013 (48 mois ; directeur : Jacques Corbeil ; codirecteur : François Laviolette). Cette thèse inclut aussi mes travaux de maîtrise de septembre 2008 à décembre 2009 (16 mois ; directeur : Jacques Corbeil ; codirecteur : Mario Marchand) puisque ayant fait un passage accéléré au doctorat, je n'ai pas eu l'occasion d'en discuter dans un mémoire.

Les 4 articles inclus dans cette thèse sont

- *HIV-1 coreceptor usage prediction without multiple alignments : an application of string kernels* paru dans le journal *Retrovirology* en 2008 (chapitre 5),

- *Ray : Simultaneous Assembly of Reads from a Mix of High-Throughput Sequencing Technologies* publié dans le journal *Journal of Computational Biology* en 2010 (chapitre 6),

- *Ray Meta : scalable de novo metagenome assembly and profiling* publié dans le journal *Genome Biology* en 2012 (chapitre 7) et

- *Human analysts at superhuman scales : what has friendly software to do ?* accepté pour publication dans le journal *Big Data* en 2013 (chapitre 8).

Les travaux présentés dans cette thèse ont été financés par les Instituts de recherche en santé du Canada :

- une Bourse d'études supérieures du Canada Frederick Banting et Charles Best à la maîtrise (# 200902CGM-204212-172830, septembre 2009–août 2010)
- et une Bourse d'études supérieures du Canada Frederick Banting et Charles Best au doctorat (# 200910GSD-226209-172830, septembre 2010–août 2013).

# Introduction

L'automatisation est ce qui conduit les technologies à être pratiques pour un nombre important de personnes. En sciences de la vie, l'automatisation permet d'accélérer les découvertes. De plus, l'automatisation augmente la reproductibilité des processus scientifiques. En génomique, 3 principales composantes technologiques guident les avancées. La première composante est le séquençage à haut débit de l'ADN [55, 273, 272, 205, 90, 113]. Le séquençage de l'ADN permet de lire le contenu de l'ADN. Les systèmes de séquençage sont largement automatisés et permettent de décoder d'immenses quantités de données. La deuxième composante technologique en génomique est l'approvisionnement automatique de ressources informatiques – tant les superordinateurs [166] que l'informatique infonuagique (*cloud computing*) [1, 263, 161, 281, 2, 76, 3, 4, 140]. Plusieurs termes ont été proposés pour décrire de tels systèmes avant que le terme *cloud computing* soit populaire – *utility computing* et *grid computing* [88, 89]. L'informatique dans les nuages permet d'abstraire la façon dont sont approvisionnées les ressources de calcul. La troisième composante importante dans ce triangle est la disponibilité de logiciels adaptatifs pour la génomique. Un logiciel est dit adaptatif s'il peut s'adapter de façon à utiliser les ressources mises à sa disposition. Le terme anglais utilisé pour l'adaptativité est la *scalability*.

Pour résumer, les données concernant un phénomène ou une expérience sont obtenues par séquençage de l'ADN. Ces données sont ensuite entreposées sur des ressources de stockage dans les nuages où ces ressources sont utilisées à l'heure. Les ressources informatiques ne sont pas spécifiques à la biologie. Mais des logiciels spécifiques pour accomplir les tâches de calcul du domaine de la biologie sont requis pour découvrir des informations utiles à partir des données.

Un autre critère important est celui de la portabilité. Un logiciel est portable s'il peut fonctionner sur différentes combinaisons d'architectures, plateformes, et de systèmes d'exploitation. Ce critère est important puisque le même logiciel sera en général utilisé

sur plusieurs jeux de données en utilisant plusieurs systèmes de calcul. Par exemple, pendant mon doctorat, j'ai eu l'occasion d'analyser des données en utilisant un Sun Constellation (colosse à Calcul Québec, Université Laval), un SGI Rackable (Mammouth Parallèle II à Calcul Québec, Université de Sherbrooke), des IBM Blue Gene/Q (Mira au *Argonne Leadership Computing Facility, Argonne National Laboratory* et bgq à SciNet), un Cray XK6 (Titan au *Oak Ridge Leadership Computing Facility, Oak Ridge National Laboratory*), et un IBM iDataPlex (guillimin à Calcul Québec, Université McGill). Bien sûr, tous ces noms ont l'air complexes, mais pour obtenir la portabilité désirée, il suffit d'utiliser une interface commune à tous ces systèmes.

Pendant mon doctorat, j'ai investi mes efforts dans la création de logiciels pour faciliter le travail des biologistes. Ces logiciels sont portables grâce à l'utilisation de l'interface de passage de messages ou MPI pour *Message Passing Interface* [87, 86, 102] – l'interface commune à la plupart des systèmes de calcul. Dans la prochaine section, la problématique initiale est présentée.

# Problématique

La capacité à générer des données numériques pour un projet de recherche de biologie surpasse de loin la capacité à analyser ces données [176, 183]. Plusieurs facteurs contribuent à cette situation : un d'entre eux est que les biologistes ne veulent pas faire d'informatique [221, 220]. De plus, le coût pour obtenir des séquences est beaucoup plus bas que le prix pour analyser les données, le coût de l'infrastructure informatique, ou encore le coût de la main-d'œuvre [176]. Lincoln Stein – un chercheur en bio-informatique – a partagé une idée pendant la conférence *O'Reilly Open Bioinformatics Conference* de 2002. Il a mentionné que la bio-informatique était trop fragmentée et qu'il y avait trop de formats. Sa vision était de créer une nation de bio-informatique au lieu d'avoir plusieurs villes incompatibles. Plus de dix années plus tard, la communauté bio-informatique est encore loin de l'utopie décrite par Lincoln Stein en 2002 [280] intitulée *Creating a bioinformatics nation*. Aujourd'hui, les espoirs pour réduire l'intensité du goulot sont diversifiés et incluent l'informatique infonuagique (*cloud computing*), l'interface graphique magique (*push-button apps*) et des changements dans les cursus universitaires [220]. L'utilisation de l'informatique dans les nuages (par exemple, en utilisant des systèmes de calcul standardisés appelés les superordinateurs) permet de standardiser les pièces (formats de données et de métadonnées) utilisées en génomique. Donc, la première problématique est qu'il y a beaucoup de données et que les outils dis-

ponibles pour les analyser se doivent de s'adapter. Une autre problématique est que les régions répétées dans les génomes sont souvent plus longues que les séquences obtenues par séquençage de l'ADN. Cette problématique est plus technique et vise le processus d'assemblage de génomes mis en place dans les assembleurs. L'assemblage de génomes consiste à construire un casse-tête à partir de petites pièces qui sont appelées les *lectures* d'ADN (en anglais : *DNA reads*).

La troisième problématique rencontrée est qu'un graphe complet de communication produit des latences très élevées sur certains types de réseaux. En effet, les systèmes haut de gamme, comme le IBM Blue Gene/Q ou encore le Cray XE6, disposent d'une réseautique à très haute performance entre les différents noeuds de calcul. Il existe aussi un standard moins dispendieux appelé *Infiniband* pour construire des superordinateurs qui permettent des opérations de communication avec une latence relativement basse.

## Hypothèses

Il est possible de distribuer un calcul d'assemblage *de novo* de génomes en utilisant le passage de messages [87] et des acteurs [5]. Un acteur est défini par une adresse et un comportement. Le calcul distribué permet d'analyser de plus gros ensembles de données en moins de temps. Cependant, une quantité trop élevée de messages peut avoir un impact sur le temps requis par un calcul.

Pour ma thèse, j'ai principalement posé trois hypothèses. L'hypothèse que j'ai posée initialement était qu'il est possible d'assembler un génome en faisant une exploration parallèle dans un immense graphe distribué. La deuxième hypothèse est que les séquences en paires permettent de traverser certaines régions répétées. La dernière hypothèse est qu'il existe une méthode pour réassigner des routes de transport alternatives pour tous les messages circulant pendant un calcul afin de réduire la latence pour les systèmes de calcul qui ne sont pas équipés d'une réseautique ayant une performance suffisante.

## Objectifs

L'objectif principal de cette thèse était de créer des une méthode massivement parallèle pour l'assemblage *de novo* de séquences d'ADN.

Voici les objectifs de cette thèse :

- Objectif général : créer une plateforme de programmation par passage de messages ;
  - Objectif spécifique : implémenter et tester différents graphes pour le calcul de routes ;
  - Objectif spécifique : explorer les façons possibles pour agglomérer des messages automatiquement ;
  - Objectif spécifique : inventer une architecture modulaire pour enregistrer des composantes distribuées ;
- Objectif général : créer un assembleur de génomes distribué ;
  - Objectif spécifique : construire un sous-graphe du graphe de de Bruijn de manière distribuée ;
  - Objectif spécifique : établir une procédure automatique pour traiter les séquences en paires ;
  - Objectif spécifique : évaluer les stratégies permises pour le partage de séquences nucléiques entre les processus ;
- Objectif général : créer un visualiseur de graphes génomiques interactif ;
  - Objectif spécifique : définir les types de données supportées et leurs formats ;
  - Objectif spécifique : programmer une interface visuelle pour naviguer les graphes de génomes.

## Résumé des contributions

J'ai inclus mes contributions de ma maîtrise puisque j'ai fait un passage accéléré au doctorat. Les contributions de ma maîtrise maîtrise (septembre 2008 à décembre 2009) sont :

- j'ai co-inventé le noyau à segments distants pour prédire le corécepteur utilisé par le virus VIH-1 et
- j'ai été le premier auteur d'un article intitulé *HIV-1 coreceptor usage prediction without multiple alignments : an application of string kernels* et publié dans le journal *Retrovirology* en 2008 (facteur d'impact : 6.47).

Les contributions de mon doctorat (janvier 2010 à avril 2014) sont,

- j'ai fait la conception et programmé RayPlatform, une librairie pour faire des logiciels massivement parallèles,
- j'ai programmé les modules pour Ray pour plusieurs types d'analyse,
- j'ai été le premier auteur d'un article dans le journal *Journal of Computational Biology* en 2010 (facteur d'impact : 1.55) intitulé *Ray : simultaneous assembly of reads from a mix of high-throughput sequencing technologies* et
- j'ai été également le premier auteur d'un article intitulé *Ray Meta : scalable de novo metagenome assembly and profiling* dans le journal *Genome Biology* en 2012 (facteur d'impact : 9.04),
- j'ai aussi co-inventé le modèle hybride de programmation appelé minis rangs,
- j'ai ajouté le support pour les minis rangs dans RayPlatform et
- j'ai écrit les plans et programmé le navigateur infonuagique Ray Cloud Browser.

Mes contributions consistent donc en la publication d'articles de qualité sur des nouvelles méthodes computationnelles en bio-informatique, en la programmation de ces méthodes dans des logiciels informatiques qui sont libres, et à des collaborations fructueuses qui ont menées à des publications. Une liste de mes publications est disponible sur http://boisvert.info/publications.html.

## Survol du contenu

Cette thèse débute par une introduction (le présent chapitre).

Le chapitre 1 fait une revue de la littérature sur les technologies de séquençage de l'ADN. Une liste de mes collaborations en génomique est présentée. De plus, ce chapitre survole très brièvement la vaste science qu'est la méta-génomique.

Le chapitre 2 apporte des concepts sur l'analyse de séquences et la bio-informatique. À la fin de ce chapitre, des concepts généraux de l'apprentissage automatique sont présentés.

Le chapitre 3 introduit le concept de messages en informatique, le passage de messages et l'interface de passage de messages. Ce chapitre apporte des informations sur le calcul à haute performance, et sur le cadriciel RayPlatform.

Le chapitre 4 introduit le graphe de de Bruijn, et discute de la théorie des graphes et les graphes importants pour les travaux décrits dans cette thèse.

Le chapitre 5 contient intégralement un article que j'ai publié en 2008 dans le journal *Retrovirology* sur l'apprentissage automatique pour prédire le corécepteur utilisé par le virus de l'immunodéficience humaine (VIH) de type 1 (VIH-1).

Le chapitre 6 contient un article que j'ai publié dans le journal *Journal of Computational Biology* en 2010. Ce travail décrit une méthode pour assembler des données de séquençage provenant de plusieurs technologies.

Le chapitre 7 contient un article que j'ai publié dans le journal *Genome Biology* en 2012 et présente une méthodologie pour l'assemblage et le profilage de méta-génomes. La méthode décrite peut analyser d'énormes jeux de données.

Le chapitre 8 contient un article qui a été accepté pour publication dans la revue *Big Data*. Je suis deuxième auteur et j'ai contribué également avec la première auteure.

Le chapitre contient une courte description des 8 articles collaboratifs auxquels j'ai contribué.

La thèse se termine par une conclusion et une bibliographie.

# Chapitre 1

# Génomique

## 1.1 Bio-informatique et biologie computationnelle

La bio-informatique est souvent vue comme l'application de l'informatique à la biologie, tandis que la biologie computationnelle est davantage un champ de recherche qui vise à développer de nouvelles méthodes computationnelles. Le professeur Pavel Pevzner de l'Université de Californie à San Diego a écrit plusieurs articles sur ce sujet [220, 221, 222]. Essentiellement, l'informatique a révolutionné la biologie et les cursus universitaires de biologie doivent inclure plus d'informatique. Je considère que mes travaux de doctorat sont classifiés dans la discipline de la biologie computationnelle.

## 1.2 Technologies de séquençage de l'acide désoxyribonucléique

La structure de l'acide désoxyribonucléique a été découverte par James D. Watson et Francis H. C. Crick [297]. Cette découverte était très importante et ils ont reçu un prix Nobel. La structure de l'ADN a aussi été étudiée dans plusieurs autres articles dans les années 1950 (voir http ://www.nature.com/nature/dna50/archive.html ). L'un de ceux-ci est l'article de James D. Watson et Francis H. C. Crick et dans celui-ci, les auteurs ont suggéré que les deux brins de l'ADN pourraient être utilisés par un mécanisme de copie du matériel génétique. Le séquençage de l'ADN a pour but de donner la séquence de nucléotides de molécules d'ADN. Cette information peut être utilisée, par exemple, pour identifier des mutations qui causent des phénotypes particuliers, ou encore pour détecter des agents infectieux dans des échantillons. Plusieurs publications ont comparé

les technologies de séquençage [235, 167, 133].

### 1.2.1 Le processus de séquençage

Plusieurs méthodes pour séquencer l'ADN se sont succédées au cours de l'évolution technologique des séquenceurs d'ADN. Un séquenceur d'ADN produit des fichiers contenant des séquences numériques d'ADN [272] (figure 1.1). Ces séquences sont des chaînes de caractères et l'alphabet utilisé est $\{A, T, C, G\}$. Le symbole $N$ est aussi utilisé pour représenter une ambiguïté.

Les types d'erreurs sont les substitutions, les insertions, et les délétions. Une substitution est présente lorsqu'un nucléotide est remplacé par un autre nucléotide différent. Le polymorphisme de simple nucléotide ou *SNP* (*Single Nucleotide Polymorphism*) est une substitution de nucléotide par un autre. Il y a une substitution entre les séquences ATGCATCGATGCTACGCAT et ATGGATCGATGCTACGCAT (la substitution est soulignée). Une insertion est produite lorsqu'une séquence est ajoutée à une position dans une séquence. Par exemple, il faut faire une insertion pour transformer la séquence ATGATCGATGCTACGCAT en la séquence ATGATCGAGCGCGCGCGCGCTGCTACGCAT (l'insertion est soulignée). Finalement, une délétion est le contraire d'une insertion. La séquence ATGATCGAGCGCGCGCGCGCTGCTACGCAT doit subir une délétion (la séquence à enlever est soulignée) pour obtenir la séquence ATGATCGATGCTACGCAT.

Dans la section suivante, la méthode de séquençage de Sanger [262, 279] est présentée.

### 1.2.2 La méthode de Sanger

Les deux premières méthodes de séquençage de l'ADN étaient celle de Maxam et Gilbert [181] et celle de Sanger [262]. Ces deux méthodes ont été publiées en 1977. L'une des premières méthodes qui a été décrite pour séquencer l'ADN était la méthode Sanger – une méthode décrite par Frederick Sanger en 1977 [262]. Frederick Sanger a reçu un prix Nobel en 1980 pour cette méthode, et il avait reçu un autre prix Nobel pour le séquençage de l'insuline en 1958. La méthode de séquençage de l'ADN de Sanger [262] utilise des terminateurs aléatoires qui bloquent la polymérisation de l'ADN par l'ADN polymérase.

Dans la méthode Sanger, 4 réactions de polymérisation sont réalisées en parallèle. Chacune des réactions contient les 4 nucléotides en concentration égale. Dans chacune

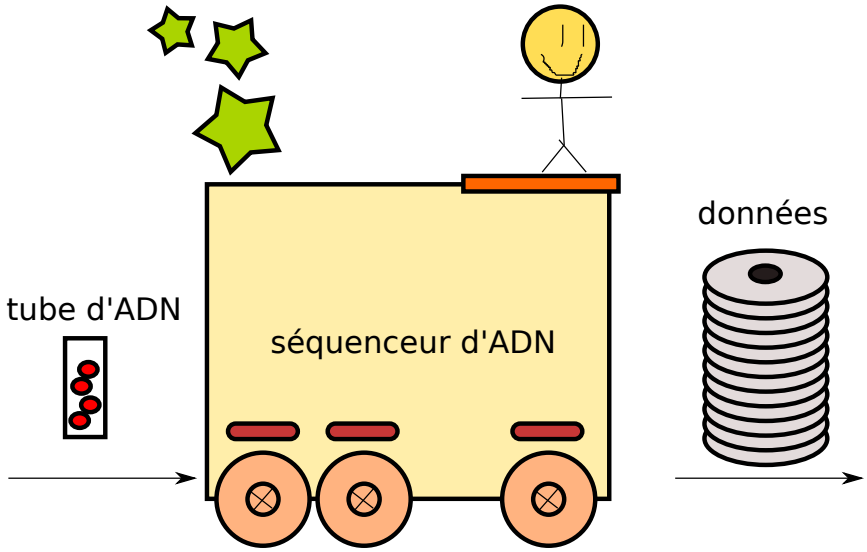


FIGURE 1.1: Un séquenceur d'ADN.

Un séquenceur est un système opéré par un humain qui a comme entrées des molécules d'ADN (ou bien des librairies contenant des molécules d'ADN) et qui a comme sorties des séquences d'ADN numérisées dans des fichiers informatiques.

d'entre elles, un des quatre nucléotides est aussi présent avec une plus faible concentration sous une forme contenant un terminateur. Au final, chaque réaction contiendra des molécules de plusieurs longueurs. Chaque réaction est déposée sur un gel, et une électrophorèse est réalisée. La méthode Sanger utilise la radioactivité pour détecter des bandes sur un gel. Ce gel contient 4 lignes de migration – une ligne par nucléotide. Puisque les molécules migrent selon leur longueur, une lecture du gel produit directement la séquence de l'ADN initial. En partant avec la bande la plus éloignée du point de départ, et en lisant les 4 lignes en direction du point de départ, le biologiste moléculaire obtient la séquence d'ADN directement. Cette méthode était longue et fastidieuse.

### 1.2.3 Automatisation avec le séquenceur 3730xl

Par la suite, la méthode de Sanger a été automatisée avec un ordinateur, l'électrophorèse capillaire, et la fluorescence [279]. La migration sur un gel de polyacrylamide a été remplacée par une séparation à l'aide d'une électrophorèse capillaire. L'avantage d'utiliser des terminateurs aléatoires est que le signal reçu de la fluorescence provient d'une seule position et non de plusieurs.

Cette méthode a été commercialisée dans le produit 3730xl de la compagnie Applied Biosystems. Cette technologie a été utilisée par le Projet Génome Humain [144, 145].

Avec cette méthode, une seule réaction (souvent appelée réaction de *BigDye Terminator*) est requise et aucune radio-activité n'est nécessaire. Les molécules d'ADN sont séparées par taille avec l'électrophorèse capillaire (habituellement 96 capillaires indépendants).

### 1.2.4 La prochaine génération

Les concepts fondamentaux pour le séquençage de l'ADN à très haut débit ont été décrits par Church en 1988 [55]. Dans ce papier, le multiplexage des cibles est un concept important. De plus, une revue de la littérature des développements les plus significatifs dans le domaine des essais génomiques parallèles a été écrite en 2006 [83]. Les principes des essais génomiques parallèles de Fan *et al.* 2006 s'appliquent aux situations suivantes : la réaction en chaîne de la polymérase (PCR) multiplexe, les puces de génotypage (essai GoldenGate ou essai avec sondes moléculaires à inversion (*MIP : molecular inversion probe*)), les puces d'hybridation comparative en génomique, l'analyse des mutations contribuant à l'expression spécifique à des allèles, et évidemment le séquençage parallèle de l'ADN. Ces principes sont aussi applicables pour le *ChIP-on-chip*, et le *ChIP-seq* [83].

### 1.2.5 Le pyroséquençage 454

La technologie 454 [247, 245, 246, 177] n'utilise pas de terminateurs aléatoires de polymérisation, contrairement à la méthode de Sanger [262, 279]. La technologie 454 utilise plutôt une enzyme appelée la luciférase pour interpréter la séquence avec de la lumière. Dans le milieu réactionnel, il y a les enzymes suivantes : la polymérase d'ADN, ATP sulfurylase, la luciférase et l'apyrase. Les substrats suivants sont aussi présents : adénosine 5' phosphosulfate (APS), les déoxynucléotides (dNTPs) et la luciférine. Premièrement, une amorce s'hybride à l'amplicon simple brin d'ADN. Lorsque l'ADN polymérase incorpore un dNTP à l'ADN, un pyrophosphate est libéré. L'enzyme ATP sulfurylase convertit le pyrophosphate en adénosine triphosphate en présence d'APS. Par la suite, la luciférase convertit la luciférine en oxyluciférine, ce qui génère de la lumière. La hauteur du pic dans le pyrogramme est proportionnelle au nombre de didéoxynucléotides incorporés [250]. La technologie 454 [177] produit des erreurs de séquençage au travers des événements de type *carry-forward* et de la polymérisation dans les régions d'homopolymères. Étant donné qu'il n'y a pas de terminateurs aléatoires dans la technologie 454, le signal devient difficile à interpréter en présence d'homopolymères. Par exemple, la technologie 454 produit un signal très similaire pour la séquence TTTTTT (6 T) et

la séquence TTTTTTT (7 T). Les produits utilisant la technologie 454 sont : Roche 454 GS, Roche 454 GS FLX, Roche 454 GS FLX+ et Roche 454 GS Junior.

Le séquençage 454 est apprécié pour les séquences longues qu'il génère. L'alignement d'une séquence de 750 nucléotides est plus spécifique que l'alignement d'une séquence ayant entre 100 et 200 nucléotides [301]

### 1.2.6 Le séquençage Illumina

Le séquençage Illumina est une approche de séquençage par synthèse. La technologie de séquençage Illumina domine le marché. Cette technologie est utilisée dans les séquenceurs de type Illumina Genome Analyzer, Illumina HiSeq et Illumina MiSeq. Différents kits de préparation des librairies sont disponibles. La librairie d'ADN est disposée sur une cellule à flux et une amplification en pont est réalisée pour augmenter la quantité de molécules afin de pouvoir les détecter. Lors du processus de séquençage, les monomères sont incorporés d'une manière similaire à la méthode Sanger – c'est-à-dire qu'il y a des terminateurs [26]. Les produits utilisant la technologie Illumina sont : Illumina Genome Analyzer, Illumina Genome Analyzer II, Illumina Genome Analyzer IIx, Illumina HiSeq 1000, Illumina HiSeq 1500, Illumina HiSeq 2000, Illumina HiSeq 2500 et le Illumina MiSeq.

#### Biais

Plusieurs articles ont souligné que la technologie Illumina produit des erreurs reproductibles [73]. Avec la technologie de séquençage de l'ADN Illumina, des substitutions de simple nucléotide peuvent parfois être causées par l'oxydation de l'ADN pendant la préparation de l'échantillon [63]. De plus, un biais concernant un des deux brins de l'ADN peut être observé lors du séquençage de l'ADN [103]. Par ailleurs, ces biais spécifiques aux brins ne proviennent pas de source biologique et peuvent créer beaucoup de sommets interconnectés dans le graphe d'assemblage [119].

Un autre biais présent dans la technologie de séquençage Illumina (ainsi que dans d'autres technologies de séquençage) est que les régions riches en guanines et cytosines ont en règle générale une profondeur de séquençage plus basse. Un appariement entre une guanine et une cytosine contient trois liaisons hydrogène. Un appariement entre une adénine et une thymine contient deux liens hydrogènes. Donc, un appariement entre une guanine et une cytosine est plus difficile à briser qu'un appariement entre une adénine et une thymine.

### 1.2.7 Autres technologies

#### Helicos

La technologie de séquençage Helicos analyse des molécules individuellement [164, 107, 209, 210]. En 2009, des chercheurs ont séquencé un génome humain pour $50 000 avec cette technologie [231]. Le séquençage direct de molécules individuelles permet d'éviter de devoir amplifier les gabarits qui seront séquencés. La compagnie Helicos Biosciences a fait faillite au mois de novembre 2012. Il s'ensuit donc que cette technologie n'est plus disponible commercialement.

**Pacific Biosciences** La technologie Pacific Biosciences analyse des molécules individuelles et en temps réel [81]. Les séquences Pacific Biosciences sont les moins biaisées, car la méthode analyse des molécules individuelles – il n'y a pas d'amplification [248]. Aussi, la technologie SMRT permet de mesurer l'activité enzymatique de l'ADN polymérase directement et est très sensible.

**Oxford Nanopore** La technologie Oxford Nanopore est en développement et utilise un pore pour y faire passer une molécule d'ADN [36]. La nature de ce qui passe dans le pore change les propriétés du pore. En particulier, le courant résiduel du pore change selon le tri-nucléotide présent dans le pore [293]. Ce type de technologie serait très avantageux car le coût de fabrication serait plus bas et pourrait être fait dans les mêmes usines qui fabriquent les microprocesseurs. Par ailleurs, une technologie utilisant des circuits de semi-conducteurs, est décrite ci-après.

**Ion Torrent** Ion Torrent, Inc. a été acheté par Life Technologies, Inc. en 2010. Les deux produits sont le *Ion Proton Sequencer* et le *Ion PGM Sequencer*. Cette technologie utilise une technologie de séquençage avec des semi-conducteurs [249]. Les gabarits d'ADN sont amplifiés sur des billes et ces billes sont déposées dans les puits d'une puce, avec exactement une bille par puits. La puce est ensuite déposée sur une grille de senseurs qui mesurent le signal (le voltage). Lorsqu'un nucléotide triphosphate est incorporé au brin séquencé, un proton est libéré et celui-ci change l'acidité du milieu (le pH). Étant donné qu'il n'y a pas de terminateurs lors de la polymérisation, la technologie produit des erreurs de séquençage dans les homopolymères.

**Moleculo** La technologie de séquençage Moleculo (commercialisé par Illumina, Inc.) permet d'obtenir des séquences dont la longueur est plusieurs milliers de nucléotides.

## 1.3 La méta-génomique

Un système possède des composantes dans sa structure, et son comportement est défini par l'agencement de ces composantes. À l'intérieur de ces composantes figurent des processus qui transforment des entrées en sorties. Ces processus définissent un comportement. Un tel système interagit avec son environnement.

La biologie des systèmes s'intéresse aux systèmes biologiques, comme les communautés d'organismes vivants. La méta-génomique consiste à analyser les génomes de communautés de microorganismes [187, 168]. En particulier, l'intérêt de la méta-génomique est de pouvoir étudier des microorganismes sans les cultiver [105, 106]. L'outil principalement utilisé pour obtenir des données en méta-génomique est le séquençage de l'ADN. Par exemple, il est possible de faire le diagnostic d'infections bactériennes avec la méta-génomique [200]. Plusieurs tâches sont associées à la méta-génomique. Une des tâches est la classification taxonomique des séquences obtenues par séquençage de l'ADN [35, 141, 42, 271]. Une autre tâche en méta-génomique est l'assemblage *de novo* de méta-génomes – c'est-à-dire de construire des séquences longues qui correspondent au matériel génétique des organismes vivant dans un environnement donné à partir des données de séquençage [149, 32, 214, 201, 202]. Avec l'assemblage *de novo*, il est possible d'analyser le contenu d'un échantillon plus efficacement [64]. En effet, des séquences plus longues sont plus spécifiques pour la méta-génomique [301]. Des séquences longues peuvent être obtenues avec l'assemblage *de novo* [48] ou avec une technologie de séquençage offrant des séquences plus longues comme les technologies Pacific Biosciences ou Moleculo. Les chercheurs étudient des environnements très diversifiés en méta-génomique, comme les océans en général [253] et l'Arctique [291] ou l'Antarctique [292], en particulier. La méta-génomique comparative consiste à comparer des méta-génomes en utilisant des mesures comme le contenu en gènes [287]. Un des objectifs principaux de la méta-génomique est de prédire les réseaux métaboliques – et donc les capacités enzymatiques – qui sont présentes dans un système pour lequel des données de séquençage de l'ADN sont disponible.

### 1.3.1 Microbiomes et méta-génomes

Le projet du génome humain [294, 145] a créé une carte génétique complète du génome humain. Cette carte est maintenant utilisée comme référentiel pour y comparer des informations obtenues pour une panoplie d'échantillons. Il est plus difficile d'obtenir une séquence de référence pour un méta-génome car les bactéries présentes ne peuvent

pas être cultivées séparément. Plus de 99% des procaryotes ne peuvent pas être cultivés en laboratoire [266, 287]. Ce fait est une des motivations derrière la méta-génomique.

# Chapitre 2

# Bio-informatique et apprentissage automatique

Le présent chapitre concerne l'analyse de séquences d'ADN avec des outils pour les analystes. Ces outils sont le plus souvent opérés par des spécialistes du domaine appelés bio-informaticiens. Donc, la bio-informatique est souvent accessible seulement aux bio-informaticiens. La raison de ce manque d'accessibilité à ces outils puissants est que la plupart d'entre eux doivent être lancés à partir de la ligne de commande. De manière générale, les personnes utilisant des données de séquençage pour des activités de recherche utilisent majoritairement Microsoft Windows. Les autres systèmes d'exploitation utilisés (en ordre décroissant) sont : Apple Mac OS X, Linux (Ubuntu, Fedora, ou autre) ou autre (OpenBSD, FreeBSD, NetBSD, Minix, ou autre). La plupart des utilisateurs de Microsoft Windows et de Apple Mac OS X n'utilisent pas du tout la ligne de commande. Pour pallier le faible nombre de bio-informaticiens (en quelque sorte les techniciens de la bio-informatique), à la quantité grandissante de séquences d'ADN, et à la baisse des prix pour obtenir des séquences d'ADN, des plateformes faciles à utiliser ont été introduites. En effet, plusieurs plateformes nouvelles permettent à un plus grand nombre de personnes (par exemple, les biologistes) d'expérimenter avec les outils de la bio-informatique sans nécessiter une connaissance approfondie de la ligne de commande. Plusieurs plateformes sont commerciales (Illumina BaseSpace, DNAnexus) alors que d'autres sont ouvertes (Galaxy [28]).

## 2.1 Types d'analyse

Principalement deux types d'analyse peuvent être réalisés sur des données de séquençage de l'ADN. Ces deux types sont les alignements (ou les analyses utilisant des alignements) et l'assemblage *de novo*. Une séquence de référence pour le génome de l'espèce étudiée est nécessaire pour aligner des séquences d'ADN. Ce n'est pas le cas pour l'assemblage *de novo*. Flicek et Birney ont écrit un très bel article d'introduction à ces deux types d'analyse [85]. En méta-génomique, l'assemblage *de novo* est souvent nécessaire puisque les génomes de références nécessaires pour aligner des séquences ne sont pas disponibles. En génétique humaine, les alignements sont beaucoup utilisés afin d'énumérer les variations génétiques présentes dans un échantillon particulier. La référence utilisée en génétique humaine est celle obtenue par le projet du génome humain [144, 145].

## 2.2 Alignement

Le reséquençage consiste à obtenir de l'information génétique pour un échantillon et à comparer cette information à une information de référence qui est considérée normale. Les différences entre les informations obtenues et le référentiel suggèrent des changements dans l'architecture génétique qui doit être validée afin d'en vérifier l'impact sur le phénotype. Le terme reséquençage est souvent associé implicitement à des méthodes d'analyse utilisant les alignements.

Plusieurs aligneurs existent : BLASR [47], SOAP2 [163], MAQ [162], BWA [158], BWA-SW [159], Bowtie [148] et Bowtie2 [147]. De plus, plusieurs aligneurs sont basés sur la librairie SeqAn [75] : RazerS [298], RazerS 3 [299] et Masai. Les structures de données et les algorithmes sont différents dans ces aligneurs. Cependant, ce sont des systèmes similaires qui alignent des lectures d'ADN (format : fastq) sur une référence (format : fasta) et produisent des alignements (format : sam [160]).

## 2.3 Construction de gabarits

La construction de gabarits (*scaffolds*) est une tâche bio-informatique. Elle consiste à ordonner et à orienter les séquences assemblées en utilisant des paires de séquences [230] ou en utilisant des cartes optiques de restriction [199]. Myers et ses collaborateurs ont proposé un algorithme vorace pour la construction de gabarits [123]. Les nouveaux outils qui permettent de construire des gabarits génomiques, comme SSPACE [29], sont des meilleures implémentations (vitesse, facilité d'utilisation).

## 2.4 Apprentissage automatique supervisé

L'apprentissage automatique supervisé est une discipline qui s'intéresse à faire apprendre des modèles à des ordinateurs à partir de jeux de données d'entraînement. Des algorithmes d'apprentissage sont capables de trouver des patrons redondants dans les données. L'apprentissage automatique consiste à faire apprendre des problèmes à une machine [284, 132]. Le plus souvent, la machine est un logiciel à l'intérieur d'un ordinateur.

Dans cette partie de ma thèse, des algorithmes d'apprentissage automatique sont présentés. De plus, les résultats (sous la forme d'article publié) que j'ai obtenus pendant ma maîtrise (j'ai fait un passage accéléré au doctorat) sont présentés dans le chapitre 5.

### 2.4.1 Mise en contexte

Beaucoup de problèmes ont des exemples avec une valeur $x \in X$ et une étiquette $y \in Y$. Donc, un objet $x \in X$ a une étiquette $y \in Y$. Voici quelques exemples (tirés du cours IFT-7002 de l'Université Laval donné par Mario Marchand) qui aide à comprendre. En diagnostic médical, $x$ est une liste de symptomes d'un patient et $y$ est sa maladie. En reconnaissance de caractères manuscrits, $x$ est une image (matrice de pixels) et $y$ est sa signification ($y \in a, b, c, \ldots, x, y, z, 0, 1, 2, \ldots, 7, 8, 9$). Un autre exemple est la classification de textes : $x$ est un texte et $y$ est le type du texte (politique, sports, météo, ou autre). Finalement, le dernier exemple est la prédiction de la valeur d'une action en bourse. $y$ est la valeur de l'action à la fermeture de la bourse et $x$ est un vecteur de 30 valeurs de cette action pour les 30 jours précédents. Une paire $(x, y) \in (X \times Y)$ s'appelle un exemple d'apprentissage.

Un problème populaire et également important est la classification binaire où les objets $X$ sont n'importe quoi et où il y a deux étiquettes possibles, c'est-à-dire que $Y = \{-1, +1\}$. En apprentissage automatique supervisé, tous les exemples ont une étiquette. En apprentissage non-supervisé, aucun des exemples n'a une étiquette. Finalement, en apprentissage semi-supervisé, une partie des exemples a une étiquette. Les algorithmes d'apprentissage automatique présupposent l'existence d'une distribution de probabilités $P$ sur les objets de l'ensemble produit par le produit cartésien des ensembles $X$ et $Y$. Par exemple, si la paire $(x, y) \in X \times Y$ est impossible, alors sa probabilité sera nulle. Dans la pratique, par contre, personne ne connaît la distribution $P$ sur l'ensemble $X \times Y$. Un échantillon de $S \subseteq (X \times Y)$ selon $P$ est souvent dispo-

nible dans les problèmes à résoudre. L'apprentissage automatique, plus spécifiquement la classification binaire en apprentissage supervisé, s'intéresse à créer une fonction d'apprentissage $A$. Cette fonction $A$ a comme but de générer un classificateur $h$ à partir d'un échantillon d'exemples. Pour obtenir le classificateur $h$ à partir de l'échantillon $S$ en utilisant la fonction d'apprentissage $A$, la notation $h = A(S)$ est utilisée. L'apprentissage automatique consiste à estimer la distribution $P$ afin de créer des classificateurs.

### 2.4.2 Noyaux

Une opération utile pour l'apprentissage automatique est la comparaison de similarité [284]. Idéalement, la relation de similarité entre $x \in X$ et $w \in X$ doit être symétrique et transitive. Un noyau est une fonction qui prend deux objets $x, w \in X$ et retourne un nombre indiquant la similarité. Un nombre grand indique une grande similarité alors que 0 indique une orthogonalité. Étant donné que nous avons dit que l'ensemble $X$ peut être n'importe quoi, il faut définir la notion de caractéristiques. Les caractéristiques d'un objet $x \in X$ sont obtenues avec la fonction $\phi : X \to \mathbb{R}^n$. Deux objets sont orthogonaux si toutes leurs dimensions dans $\phi(x)$ sont indépendantes et sans relation. Alors, la similarité calculée par un noyau $k$ associé à la fonction $\phi$ est simplement le produit scalaire $k(x, w) = \langle \phi(x), \phi(w) \rangle = \sum_{i=1}^{n} \phi_i(x) \cdot \phi_i(w)$ où chaque caractéristique est définie par $\phi_i : X \to \mathbb{R}$. La fonction $\phi$ d'un noyau $k$ peut très bien retourner un nombre infini de caractéristiques, (c'est-à-dire qu'il est possible de considérer le cas limite $n = \infty$). Dans ce cas, $k$ est calculé avec le truc du noyau qui consiste à ne pas calculer $\phi(x)$ explicitement. Ce processus consiste à calculer directement (et rapidement) $k(x, w)$ sans jamais calculer aucun $\phi(x)$ et aucun $\phi(w)$. Le truc du noyau n'est pas toujours possible pour toute fonction $\phi$, mais pour un grand nombre de cas d'intérêt en pratique, c'est possible. Pour les cas où le nombre de caractéristiques est infini, il faut que le produit scalaire converge. C'est le cas du noyau appelé le noyau RBF (*Radial Basis Function*).

Pour la biologie, le contenu en mots d'un brin d'ADN ou encore d'une protéine génère beaucoup de caractéristiques, particulièrement lorsque des tuples de mots sont considérés. Dans ce cas, le truc du noyau consiste à calculer le produit scalaire sans passer par les vecteurs explicites de caractéristiques. Ces noyaux s'appellent les noyaux à chaîne [154, 155, 114, 305, 256, 257].

### 2.4.3 Machine à vecteurs de support

Un robot logiciel populaire s'appelle la machine à vecteurs de support. L'interface entre cette machine et les exemples d'apprentissage est le noyau [132]. Cet algorithme d'apprentissage assigne à chaque exemple d'apprentissage un multiplicateur de Lagrange afin de construire un hyperplan de séparation dans l'espace des caractéristiques [61, 290]. Un des points intéressants est que la majorité du temps cette machine attribue un multiplicateur nul à la plupart des exemples, car les motifs dans les données sont redondants. Un objet $x \in X$ est classifié comme $-1$ ou comme $+1$ en regardant sa position par rapport à l'hyperplan. Un autre point intéressant est la nature très modulaire de cet algorithme. Premièrement, n'importe quel noyau peut être branché sur ce robot logiciel, ce qui permet de l'appliquer à n'importe quel jeu de données. De plus, il est possible d'adapter la fonction de régularisation et la fonction objective pour, par exemple, permettre plus d'élasticité dans les erreurs.

### 2.4.4 Le noyau à segments distants

Pendant un stage, j'ai co-inventé le noyau à segments distants et nous l'avons publié dans le journal scientifique *Retrovirology* [31].

La fonction de caractéristiques $\phi$ de noyau utilise des paires de mots peptidiques dans une protéine (ou partie de protéine). Le vecteur de caractéristiques est énorme. Pour cette raison, le truc du noyau est utilisé. Mais qu'est-ce que le truc du noyau. Cette astuce consiste à calculer un produit scalaire entre deux objets $x$ et $x'$ sans calculer explicitement $\phi(x)$ et $\phi(x')$. Pour ce faire, l'algorithme du *distant segments kernel* itère seulement sur les caractéristiques qui sont partagées entre $x$ et $x'$. Le chapitre 5 contient l'article publié dans la revue *Retrovirology*. Dans ce chapitre, l'algorithme détaillé du noyau à segments distants est fourni et une preuve de son exactitude y est aussi présente.

# Chapitre 3

# Passage de messages

## 3.1 Mise en contexte

Étant donné qu'il y a beaucoup de données en génomique, il est difficile d'entreposer l'ensemble des données dans le même endroit. Pour stocker et traiter des quantités énormes de données, ces données sont donc distribuées dans plusieurs endroits (comme par exemple des noeuds de calcul) et la communication inter-noeud est utilisée pour la transmission de données entre les noeuds, lorsque nécessaire. Le passage de messages est utilisé pour la communication.

Dans ce chapitre, une plate-forme logicielle qui utilise le passage de messages est décrite. Cette plate-forme s'appelle RayPlatform et le logiciel d'assemblage de génomes appelé Ray l'utilise. La section 3.2 décrit le passage de messages et la section 3.6 décrit la plate-forme RayPlatform.

## 3.2 Introduction au passage de messages

Un message est une information qui est transmise d'une source à une destination et contient en général aussi un sujet [116]. Un message est transporté en utilisant le passage de messages. Plusieurs moyens sont disponibles pour passer un message : par la poste, par le réseau Internet, ou encore en utilisant des pigeons voyageurs. La latence est le temps nécessaire pour effectuer le transport d'un message. Une très basse latence est désirée afin d'éviter d'attendre trop longtemps pour obtenir une réponse à un message. En effet, les dépendances de données entre les différents processus peuvent s'avérer être une source de problèmes.

Dans ce chapitre, le passage de messages est présenté. Le passage de message permet d'avoir des applications distribuées (aussi appelées applications réparties). Les messages sont échangés entre les parties en utilisant un protocole de communication [116]. Les composantes impliquées dans un système de passage de messages comprennent au moins deux processus (qui peuvent être le même), un canal de communication et évidemment un message. Les 5 étapes nécessaires pour la transmission d'un message sont la création, l'envoi, la livraison, la réception et le traitement [116]. Les étapes de création et d'envoi sont effectués par le processus source. Le canal de messagerie est responsable de la livraison. Finalement, le processus de destination est responsable des étapes de la réception et du traitement.

Les systèmes logiciels qui utilisent le passage de messages doivent respecter des conventions et utiliser des standards. Par exemple, les applications scientifiques utilisent le standard MPI (*Message Passing Interface*, standardisé par le *Message Passing Interface Forum*). Le standard MPI est décrit sous forme d'un livre et les deux dernières versions de ce standard sont MPI 2.2 [86] et MPI 3.0 [87]. Plusieurs livres sont aussi disponibles pour apprendre à l'utiliser [102]. La beauté du standard MPI est qu'il permet de créer des programmes qui utilisent la messagerie par passage de messages sans se soucier du canal de messagerie qui sera ultimement en opération.

Sur un superordinateur typique, un message envoyé d'un processus à lui-même est transmis par une simple copie de mémoire (appel à memcpy sous les systèmes POSIX). Un message envoyé d'un processus à un autre processus sur la même machine passera par le système de mémoire virtuelle du système d'exploitation qui permettra de cartographier une zone partagée de mémoire dans les espaces d'adressages [283]. Finalement, un message envoyé d'un processus à un autre processus qui est sur une machine différente passera par un réseau liant les deux ordinateurs. Différentes technologies existent, dont TCP (*Transmission Control Protocol*), Infiniband, le circuit Gemini de Cray, Inc. [270], ou encore le tore asymétrique du IBM Blue Gene/Q [51].

## 3.3 Interface de passage de messages

Le *Message Passing Interface* (MPI), ou interface de passage de messages, est un standard de programmation qui permet de passer des messages entre les processus qui s'exécutent sur des ordinateurs. Il permet d'écrire des codes sources portables en Fortran, C ou C++. Un programme implémenté avec l'interface de passage de messages pourra s'exécuter autant sur ordinateur portable que sur des milliers de processus distribués

sur un superordinateur.

Une librairie MPI implémente les routines qui permettent d'acheminer un message d'un processus à lui-même, entre deux processus qui résident dans le même espace de processus (gérés par le même système d'exploitation), ou encore entre deux processus qui sont dans des espaces de processus différents. La couche de transport des octets est abstraite et la même interface est en général utilisée pour envoyer un message au même rang (un rang est simplement un numéro donné à un processus pour qu'il puisse envoyer et recevoir des messages ; le concept de rang est défini à la section 3.3.1), à un rang sur la même machine ou encore à un rang résidant sur une autre machine.

### 3.3.1 Rang

Dans un ordinateur, plusieurs processus s'exécutent pour réaliser des tâches. Un processus est un programme en cours d'exécution et son comportement est défini par ses entrées et par le code source qui a été utilisé pour construire le programme. Un processus a un numéro de processus enregistré par le système d'exploitation et utilise des ressources. Parmi ces ressources, il y a la mémoire virtuelle, les cycles de calcul sur un processeur virtuel du système d'exploitation et les descripteurs de fichier. Dans son excellent livre sur les systèmes d'exploitation, Andrew S. Tanenbaum sépare les ressources gérées par le système d'exploitation principalement en trois catégories : processeurs, mémoire, et disques [283].

Afin d'empêcher chaque processus de corrompre la mémoire des autres, il est nécessaire de mettre en place des mécanismes de compartimentation afin d'offrir une modularité facile et sécuritaire avec protection. C'est ce principe de protection qui permet à n'importe quel processus qui s'exécute de bénéficier d'une protection de sa mémoire. Il est donc impossible pour un processus d'aller écrire ou lire dans la mémoire d'un autre processus. Pour partager de l'information, deux processus doivent s'envoyer des messages, ou encore utiliser de la mémoire partagée.

Dans le contexte de l'interface de passage de messages, chaque processus a un numéro qui lui est unique. Ce numéro est appelé le rang. Le rang est une adresse ou un nom au sens de la théorie des acteurs [112, 6, 5, 89]. Dans la théorie des acteurs, chaque acteur a un nom et un comportement. Un acteur peut faire trois actions : envoyer des messages, créer des nouveaux acteurs et mettre à jour son état [5].

### 3.3.2 Messages

Un message contient une source, une destination, un type de message et un contenu. La source et la destination sont des acteurs (*e.g.* des rangs ou des processus). Un message peut être vu comme un courriel avec les attributs suivants : un auteur, un destinataire, un sujet et un contenu. Dans MPI, les rangs s'échangent des messages. La plupart des implémentations MPI utilisent les processus du système d'exploitation pour faire exister les rangs. La figure 3.1 montre un message qui est en train de se faire transporter.

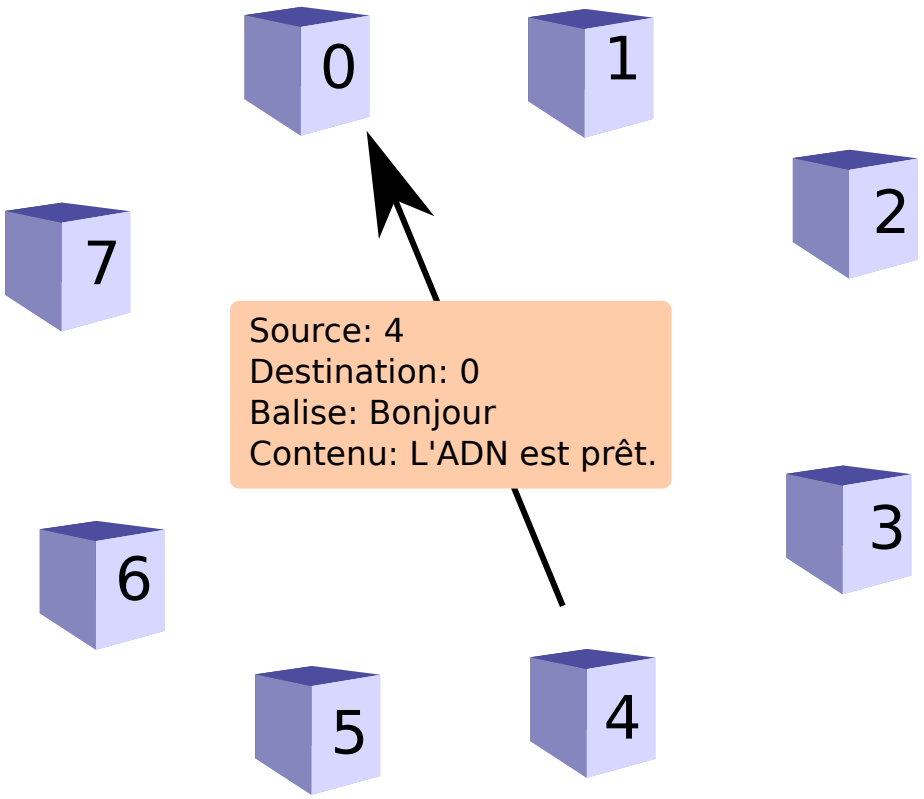


FIGURE 3.1: Un message.

Un message, formellement, possède une source, une destination, une balise et un contenu. Dans cette illustration, l'acteur portant le nom 4 envoie un message à l'acteur portant le nom 0.

### 3.3.3 Passage de messages

Le passage de messages entre les processus est naturel. Par exemple, D-Bus est un système de bus pour les messages. Ce système est utilisé par plusieurs distributions GNU/Linux. Dans ce modèle, chaque processus s'inscrit dans le répertoire central de D-Bus en fournissant un nom unique tel que « vendeur » ou encore « acheteur ». Le processus portant le nom « acheteur » peut alors envoyer un message au processus

portant le nom « vendeur » en faisant abstraction de l'endroit où ce processus est situé. Le système de bus de messages va acheminer le message de la source « acheteur » à la destination « vendeur » d'une manière ou d'une autre, et cette manière est abstraite.

Pour les applications à haute performance, l'interface de passage de message (voir la section 3.3) attribue des numéros séquentiels (les rangs) automatiquement aux processus présents dans le communicateur MPI_COMM_WORLD. Un communicateur contenant 50 processus numérotera ceux-ci de 0 à 49. Ces adresses simples sont ensuite utilisées par le système de bus de messages pour transporter les messages entre n'importe quelle paire de processus.

### 3.3.4 Librairies MPI

Il existe principalement deux implémentations de l'interface de passage de messages, soit Open-MPI [91] et MPICH [101]. Il existe beaucoup d'autres implémentations, et certaines d'entre elles sont dérivées d'autres. Quelques-unes des autres librairies MPI (liste non-exhaustive) sont DeinoMPI, Intel MPI (dérivé de MPICH), Microsoft MPI, MVAPICH2, IBM Platform MPI, FT-MPI, LA-MPI, LAM/MPI et Cray MPI. Toutes ces librairies fournissent et implémentent la même interface, mais sont disponibles sur différentes plateformes de calcul.

### 3.3.5 Création et gestion de processus

Tout d'abord, MPI_Init et MPI_Finalize sont les procédures qui doivent être appelées respectivement au début et à la fin d'un programme MPI. Ces deux procédures sont obligatoires afin d'initialiser l'environnement MPI. L'initialisation d'un tel environnement est décrite dans le chapitre 8 du standard MPI 3.0. Dans le chapitre 10 du standard MPI 3.0, la création et la gestion des processus est décrite en profondeur. Par exemple, la fonction MPI_Comm_spawn permet de créer des processus dynamiquement. En d'autres termes, il est possible d'ajouter des rangs à un calcul qui est déjà commencé. Cependant, MPI_Finalize (utilisé pour terminer un rang) est une procédure collective et donc il est impossible pour un rang de quitter (il n'y a pas de procédure qui fait le contraire de ce que fait MPI_Comm_spawn) sans que les autres rangs le fassent aussi. À cause de cette restriction, il est impossible d'enlever des ordinateurs pendant le déroulement d'un calcul.

### 3.3.6 Communication point à point (à deux côtés)

À la page 11 de MPI 3.0 [87], les termes définis sont les suivants. Une procédure est **non bloquante** si la procédure peut retourner avant d'avoir complétée. L'appel initial à la procédure démarre l'opération. Par contre, MPI_Test peut retourner vrai même si le message n'a pas encore atteint la destination. Une communication complète lorsque toutes les opérations participantes complètent. Une procédure est **bloquante** lorsqu'elle retourne seulement lorsque les ressources fournies peuvent être ré-utilisées. Une procédure est **locale** lorsqu'elle dépend seulement du processus courant. Alternativement, une procédure est **non locale** lorsqu'elle dépend sur l'exécution de procédures sur d'autres processus.

Les opérations point à point sont décrites dans le chapitre 3 du standard MPI 3.0 [87]. La communication point à point à deux côtés implique deux parties : une source et une destination. La fonction MPI_Send permet d'envoyer un message. La fonction MPI_Send est en principe une opération bloquante. Une opération est dite bloquante si le retour est effectué que lorsque l'opération est complétée. En pratique, les implémentations MPI ne sont pas aussi simples, car elles utilisent des stratégies pour optimiser les opérations.

Selon le type de média utilisé pour le transport des octets et le seuil d'empressement (*eagerness*), la librairie MPI utilisera une stratégie par empressement ou une stratégie par rendez-vous. Le seuil d'empressement est un nombre d'octets. Le rendez-vous est utilisé lorsque la taille du message dépasse le seuil d'empressement. Dans le cas contraire, l'empressement est utilisé.

Le standard MPI décrit plusieurs fonctions pour la communication point à point à la page 23 de MPI 3.0 [87]. Une fonction populaire est MPI_Isend qui permet d'envoyer un message d'un rang à un autre rang sans attendre la complétion. Cette procédure est donc non bloquante, mais le code doit vérifier que l'opération a complété avec un appel à MPI_Test.

Lorsqu'une source envoie un message à une destination, la destination doit recevoir le message. La réception bloquante peut être faite avec la procédure MPI_Recv. L'équivalent non bloquant existe pour la réception de messages : MPI_Irecv.

Si le nombre de messages (et le comportement de traitement associé) est connu *a priori*, alors MPI_Send et MPI_Recv sont suffisants. Il existe une troisième catégorie de procédures pour questionner le communicateur. La fonction MPI_Iprobe retourne

immédiatement le contrôle au code appelant et permet de savoir si des messages sont disponibles pour être reçu.

### 3.3.7 Communication à un côté

Les capacités d'accès à la mémoire distante (*Remote Memory Access*, ou RMA) permettent un accès par un processus à la mémoire d'un autre processus. Un processus peut aussi utiliser le RDMA (pour *Remote Direct Memory Access*) pour accéder directement à la mémoire distante. Le chapitre 11 du standard MPI 3.0 [87] définit des procédures pour les accès distants à la mémoire. Pour ce faire, un processus désirant rendre disponible une partie de sa mémoire doit utiliser MPI_Win_create pour créer une fenêtre de mémoire qui sera visible par les autres processus. Par la suite, des opérations peuvent être faites : MPI_Put, MPI_Get et MPI_Accumulate. Ces procédures ont des équivalents qui prennent aussi une requête (du type MPI_Request) en argument afin de pouvoir tester la complétion de l'appel. Les versions qui prennent une requête de type MPI_Request sont donc asynchrones.

### 3.3.8 Opérations collectives

Une opération collective doit être appelée par tous les rangs d'un communicateur (chapitre 6 du livre MPI 3.0) pour que celle-ci complète. Les opérations collectives sont décrites dans le chapitre 5 du standard MPI 3.0 [87]. Une procédure collective nécessite que tous les rangs dans un groupe de communication l'utilisent. MPI_Alltoall est un exemple de procédure collective. Si au moins un des rangs d'un groupe de communication ne réalise pas l'appel à une procédure collective alors que les autres rangs réalisent l'appel, alors tous les rangs vont attendre après le retardataire. Si le rang qui n'a pas fait l'appel ne le fait pas, alors les rangs MPI ne vont jamais s'arrêter. Une telle situation est très problématique (et donc à éviter autant que possible) puisque l'utilisateur du logiciel peut penser que le logiciel est en train de calculer quelque chose.

### 3.3.9 Entrées/sorties parallèles

Le standard MPI permet de lire et d'écrire des fichiers en parallèle. Le but est de maximiser la vitesse à laquelle les entrées et sorties sont réalisées sur un système de stockage. Le chapitre 13 du standard MPI 3.0 [87] décrit les procédures pour réaliser des entrées et sorties parallèles. Toutes les procédures dans *MPI I/O* sont des procédures collectives. Pour opérer sur un fichier avec les entrées et sorties parallèles, il faut

utiliser le type MPI_File. L'ouverture se fait avec MPI_File_open, la lecture se fait avec MPI_File_read et l'écriture avec MPI_File_write. La procédure MPI_File_close ferme un fichier de type MPI_File. Par défaut, toutes les opérations sur un objet MPI_File se font sur une vue à partir du décalage 0. La procédure MPI_File_set_view permet de changer le décalage en vigueur pour la vue sur le fichier. Une pratique courante est d'attribuer une vue différente à chaque rang afin qu'ils puissent tous écrire dans le même fichier, mais à des endroits différents.

## 3.4 Micronoyaux

Un *micronoyau* est un noyau dont la quantité de code opérant en mode noyau est très petite. La majorité des fonctionnalités est implémentée en mode utilisateur en utilisant le passage de messages pour intéragir avec la petite partie qui est en mode noyau. Donc, le préfixe micro dans micronoyau fait référence à la très petite taille du code en mode noyau. Donc, le passage de messages est aussi utilisé dans les systèmes d'exploitation.

Minix est un micronoyau qui utilise le passage de messages pour sécuriser tout le système [109]. En effet, lorsqu'un pilote de logiciel s'exécute à l'intérieur d'un processus et communique par passage de messages avec le micronoyau, le code de ce pilote ne peut pas faire des accès à la mémoire qui n'est pas dans sa mémoire virtuelle puisque ce processus s'exécute en mode utilisateur. En particulier, il est intéressant de constater que Minix est un noyau qui utilise le passage de messages pour permettre à ces processus de communiquer ensemble. Chaque pilote logiciel est un processus avec son propre espace d'adressage. De plus, les services du noyau sont aussi des processus.

Par opposition, Linux est un noyau qui abstrait un ordinateur pour les logiciels qui s'exécutent sur celui-ci [286]. Lorsqu'un nouveau matériel est ajouté à un ordinateur, il est nécessaire de charger un pilote logiciel pour l'utiliser. Dans Linux, le code d'un pilote est ajouté dynamiquement au code du reste du noyau (qui s'exécute en mode noyau). Cela peut être très problématique si le nouveau pilote contient du code malicieux qui pourra infecter tout le système.

Une conception par passage de messages est plus robuste qu'une conception de type monolithique. Donc, le passage de messages ne s'appliquent pas seulement aux applications scientifiques à haute performance, mais aussi à d'autres systèmes informatiques plus généraux.

## 3.5 Logiciels adaptatifs

Le calcul à haute performance concerne les logiciels qui peuvent être déployés sur les superordinateurs en utilisant le passage de messages. Les superordinateurs sont, en général, des systèmes distribués dans lesquels figurent plusieurs ordinateurs qui sont connectés avec une réseautique à haute performance (figure 3.2). Plusieurs concepts permettent de créer des logiciels qui sont robustes et à l'échelle.

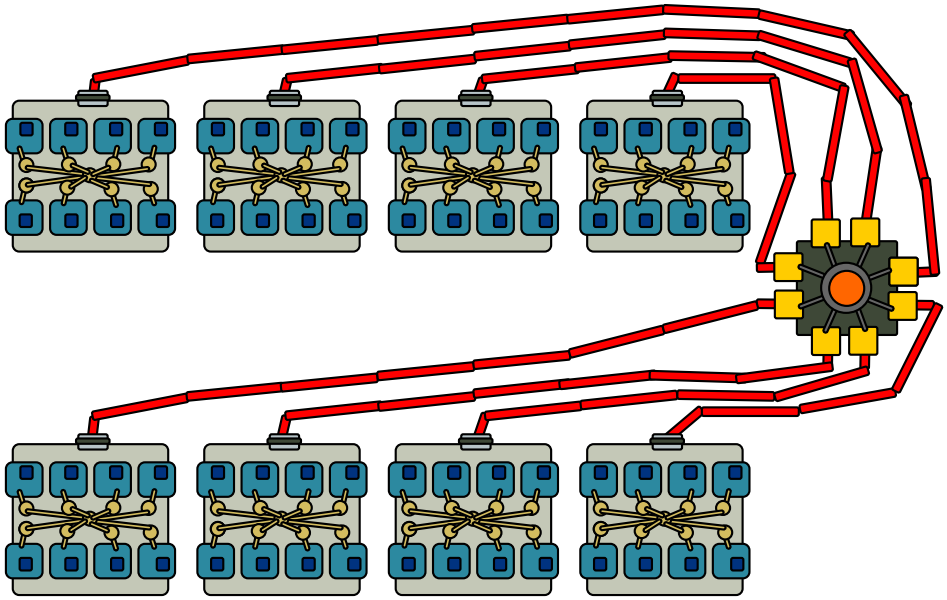

FIGURE 3.2: Une illustration simplifiée d'un superordinateur.

Le superordinateur contient 8 ordinateurs connectés avec des fils rouges sur un routeur (à droite). Chaque ordinateur a 8 unités de calcul. Ce système a donc en tout 64 unités de calcul.

Un logiciel est adaptatif s'il peut s'adapter aux ressources qui lui sont fournies afin, par exemple, de pouvoir s'exécuter en moins de temps. Il existe principalement deux façon de réaliser une mise à l'échelle : la mise à l'échelle horizontale (*scale out*) et la mise à l'échelle verticale (*scale up*). La mise à l'échelle horizontale consiste à ajouter des ordinateurs à une application distribuée. C'est le modèle principal utilisé par les superordinateurs (avec, par exemple, une réseautique Infiniband) et par les compagnies comme Facebook, Google et Amazon. La mise à l'échelle verticale, au contraire, vise à augmenter la capacité des ordinateurs existants. Ajouter un disque à un ordinateur de façon à augmenter sa capacité de stockage ou encore ajouter des processeurs à un ordinateur sont deux cas de mise à l'échelle verticale. Un autre cas de mise à l'échelle verticale consiste à remplacer les processeurs existants par d'autres plus performants.

### 3.5.1 Types de parallélisme

En calcul parallèle, principalement deux types de parallélisme sont utilisés. Les autres types sont des variantes ou bien une combinaison des deux types de base. Le premier type consiste à utiliser plusieurs fils d'exécution à l'intérieur du même processus de façon à exploiter le matériel de l'ordinateur [283]. Ce type de parallélisme est relativement facile à implémenter, mais ne permet pas un isolement complet des composantes parallèles. De plus, l'utilisation de plusieurs fils d'exécution à l'intérieur d'un processus ne permet pas d'effectuer des calculs distribués sur plus qu'une machine. Le deuxième type de parallélisme est le passage de messages. Avec le passage de messages, les différentes composantes possèdent des données privées, et des messages sont échangés pour partager de l'information. Le passage de messages permet de créer des logiciels qui s'exécutent sur plusieurs ordinateurs en même temps. De plus, les logiciels utilisant le passage de messages sont plus robustes [116].

### 3.5.2 Granularité

Un accès à la mémoire est plus rapide que l'envoi d'un message, la livraison et la réception de la réponse. Clairement, il faut donner au processeur quelque chose à faire pendant que la réponse à un message se fait attendre. La stratégie la plus gagnante est d'entrelacer la communication et le calcul. Dans un programme parallèle, il est fréquent que la communication et le calcul soient entrelacés. Dans ce contexte, la granularité est un terme utilisé pour décrire cet entrelacement. Une granularité fine indique que la quantité de calcul entre les communications est petite. Une granularité grossière signifie que la quantité de calcul entre les communications est grande. La granularité de Ray est fine [30, 32].

### 3.5.3 Amortissement

L'amortissement est une méthode qui consiste à étaler les opérations à l'intérieur de plusieurs appels [139]. De cette façon, le travail est accompli sur une plus longue durée, mais l'impact de chaque parcelle accomplie est minimal. Dans le contexte plus large du calcul parallèle, l'amortissement est souvent réalisé en utilisant une granularité fine. Dans la boucle principale, les courtes séquences d'instructions sont introduites afin de répartir des longs calculs sur un nombre élevé d'itérations. En l'absence d'amortissement, un rang ne peut pas recevoir des messages rapidement, car il doit attendre que son long calcul se termine.

Les concepts de passage de messages, de granularité et d'amortissement sont donc étroitement reliés. Une recette qui fonctionne bien est d'utiliser des petits messages, une granularité fine et un amortissement sur de longues périodes.

### 3.5.4 Structure d'un programme distribué

Une façon d'écrire un programme parallèle par passage de messages est de commencer avec une boucle principale. Dans cette boucle, les messages sont reçus, traités, puis le rang fait des opérations en fonction de son état et finalement les messages sont envoyés. La figure 3.3 illustre une telle boucle dans un programme parallèle.

```
while(1) {
    recevoirLesMessages();  // MPI_Iprobe, MPI_Recv
    traiterLesMessages();  //
    avancerLeTravail();      //
    envoyerLesMessages();     //  MPI_Isend
}
```

FIGURE 3.3: Boucle principale d'un programme distribué.

Dans la boucle principale d'un programme distribué, les messages sont reçus et traités. Ensuite, le rang avance son travail. Finalement, il envoie ses messages.

### 3.5.5 Modèle de communication utilisé par Ray

Ray est le logiciel sur lequel j'ai travaillé. Le modèle de communication utilisé est similaire à celui utilisé par d'autres logiciels.

Quel modèle de communication est utilisé par le logiciel Ray ? Premièrement, les messages contiennent un nombre maximal d'octets qui est 4000. Ray utilise une boucle comme dans la figure 3.3. NAMD est un logiciel de dynamique moléculaire [226]. Ce logiciel utilise le cadre de développement appelé CHARM++ [134]. Ray est un assembleur de génomes [30, 32]. Ray utilise le cadre de développement appelé RayPlatform. Le modèle de communication de RayPlatform est très similaire à celui de CHARM++. Les messages sont envoyés avec la procédure MPI_Isend de manière asynchrone. Le suivi et la vérification de la complétion de l'opération sont faits en utilisant l'objet MPI_Request fourni par MPI_Isend. Dans Ray, l'échange de messages n'est pas planifié. Donc, un rang ne peut pas utiliser directement MPI_Recv puisqu'il ne sait pas si un message est disponible pour être reçu. Il s'ensuit donc qu'une troisième procédure (la première est MPI_Isend et la deuxième est MPI_Recv) est requise. Pour vérifier si

un message est disponible, un rang doit appeler la procédure MPI_Iprobe. MPI_Probe est bloquante, donc dans l'absence de message, MPI_Probe attend qu'un message correspondant arrive. Donc, pour la communication point à point, les procédures utilisées sont : MPI_Isend, MPI_Iprobe et MPI_Recv. Le logiciel Ray utilise également *MPI I/O* pour écrire certains fichiers de résultat.

## 3.6 Engin de calcul RayPlatform

Le calcul est l'outil principal pour les sciences de la donnée [180, 74, 264]. Le calcul permet en effet de transformer des données en connaissances. L'informatique infonuagique facilite les calculs et la distribution des données et résultats. De manière similaire, les superordinateurs facilitent le processus scientifique – du moins quand le calcul est de la partie – en fournissant une plateforme où les tâches se font attribuer des identifiants.

En génomique, il existe des librairies – SeqAn [75] et NGS++ [208] pour ne nommer que celles-ci – pour le calcul à haute performance (*e.g.* en C++), mais celles-ci ne sont pas parallèles.

Le domaine de la simulation de modèles tridimensionnels bénéficie de plusieurs librairies parallèles. Par exemple, PETSc [16] et Trilinos [111] sont des librairies parallèles qui permettent de résoudre des problèmes d'éléments finis. CHARM++ est une librairie avancée pour créer des logiciels distribués adaptatifs dans le domaine de la dynamique moléculaire [134]. CHARM++ permet, entre autres, de faire de la migration dynamique de tâches en cours de calcul de façon à balancer la charge computationnelle sur les ressources calculatoires disponibles.

Pendant mon doctorat, j'ai développé RayPlatform, une librairie qui permet de faire des logiciels parallèles. RayPlatform est une librairie C++ pour faciliter la création de programmes massivement parallèles en utilisant l'interface de passage de messages (MPI). Le standard de programmation utilisé dans RayPlatform est le C++ 1998. RayPlatform n'est pas un logiciel, mais plutôt une librairie pour créer des logiciels. RayPlatform est utilisé comme cadre de développement de logiciels. Donc, RayPlatform est un cadriciel. RayPlatform est de moins grande envergure que CHARM++ et n'est pas spécifique à la génomique. Selon les standards de [93], RayPlatform est relativement bien conçu en incorporant beaucoup de motifs de conception. .

Dans la figure 3.4, la composition de classes pour le module SpuriousSeedAnnihilator de Ray est montrée. En utilisant les interfaces de RayPlatform (CorePlugin, TaskCreator,

et Worker), ce module est compatible avec l'engin d'exécution de RayPlatform. Ces interfaces permettent de développer plus rapidement des logiciels qui sont adaptatifs – c'est-à-dire qui peuvent exploiter les ressources matérielles disponibles.

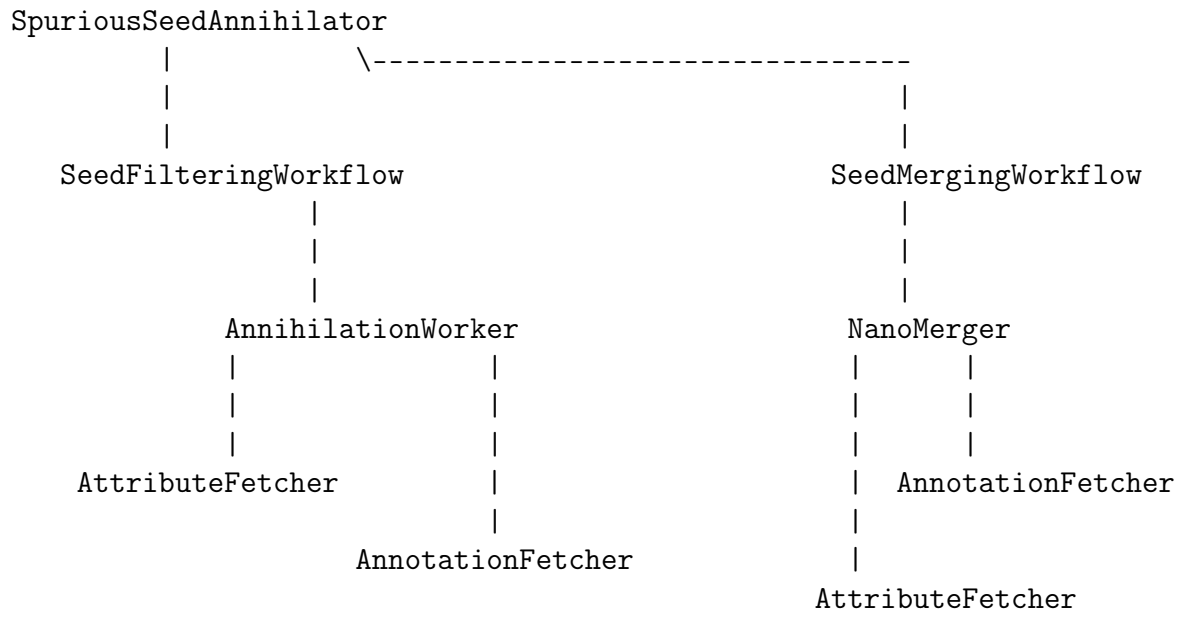


FIGURE 3.4: Classes du module SpuriousSeedAnnihilator.

La classe SpuriousSeedAnnihilator implémente l'interface CorePlugin de RayPlatform. Les classes SeedFilteringWorkflow et SeedMergingWorkflow implémentent l'interface TaskCreator de RayPlatform. Les classes AnnihilationWorker et NanoMerger implémentent l'interface Worker de RayPlatform.

### 3.6.1 Services

Les services offerts par RayPlatform sont la communication, l'engin de calcul distribué, la cryptographie, les routines gestionnaires, la gestion de la mémoire, l'architecture modulaire d'extensions, les profileurs en temps réel, le sous-système de distribution de messages par routage virtuel, le planificateur de tâches, et plusieurs structures de données importantes utilisées pour construire l'engin de stockage distribué pour les données génomiques dans Ray.

RayPlatform est générique, il ne contient rien de spécifique à la génomique ou à Ray. L'engin de stockage distribué utilise une table de hachage distribuée avec adressage ouvert [139].

### 3.6.2 Utiliser RayPlatform

Pour utiliser RayPlatform, il faut inclure les en-têtes C++ (des fichiers .h) et lier les fichiers objets avec la librairie libRayPlatform.so (ou libRayPlatform.a). RayPlatform est un projet libre et ouvert. Le site web du projet est
https ://github.com/sebhtml/RayPlatform.

### 3.6.3 Découpage

RayPlatform est une librairie qui a bourgeonné à partir de Ray. RayPlatform est un projet séparé de Ray depuis janvier 2012, alors que le projet Ray a débuté deux années avant, soit en janvier 2010. L'idée de RayPlatform est de soutenir une organisation des composantes logicielles d'un logiciel distribué. Donc, Ray est un paquet d'extensions par-dessus RayPlatform. Les plans de conception de RayPlatform n'existent pas, mais son code source est bien documenté.

### 3.6.4 Composantes du système

Il est important que les composantes logicielles soient orthogonales afin de pouvoir les faire évoluer de manière indépendante. RayPlatform offre une architecture logicielle pour créer des logiciels massivement distribués. Un logiciel utilisant RayPlatform s'exécute avec un superviseur léger qui possède des états. Chaque module doit s'enregistrer avec l'interface de programmation de RayPlatform. Un module implémente l'interface CorePlugin et peut enregistrer trois types de ressources : les modes maîtres (MasterMode), les modes esclaves (SlaveMode), et les balises de message (MessageTag). Un module peut exporter ses ressources allouées pour que d'autres modules les utilisent. De plus, il est aussi possible d'assigner un comportement à un mode esclave, à un mode maître ou encore à une balise de message de façon à définir le comportement à utiliser lorsque l'engin est dans un de ces états.

### 3.6.5 Modules d'extension

Un logiciel utilisant RayPlatform doit définir des extensions RayPlatform. Ces extensions sont branchées dynamiquement sur l'engin de calcul RayPlatform et sont exécutées de manière distribuée. L'interface à utiliser est CorePlugin. Cette interface permet d'enregistrer un module d'extension.

#### Adaptateurs

Premièrement, une extension doit demander une clé d'accès à l'engin RayPlatform. Cette clé doit être utilisée pour chacune des opérations faites avec l'engin. Ceci aide à faire des programmes plus robustes.

Un module d'extension peut enregistrer plusieurs clés de gestion (*handles*). Ces clés permettent d'enregistrer des comportements sur la plateforme. Les types de clé de gestion supportés sont les clés de balise de message, les clés de mode d'esclave et les clés de mode de maître. Pour chaque clé, il est permis d'enregistrer un gestionnaire d'événement pour indiquer à l'engin RayPlatform comment il doit réagir lorsque, par exemple, il y a un événement de réception pour un message ayant une balise donnée.

La complexité résultante d'un logiciel utilisant RayPlatform émerge de l'interaction des différents gestionnaires. Par exemple, l'arrivée d'un message provoque une cascade d'événements. Ces événements peuvent être l'émission d'un message vers la source, le changement de mode esclave ou de mode maître de la destination, ou autres procédures impliquant des modifications aux modes d'esclave ou aux modes de maître. Cette approche est basée sur le formalisme des acteurs [5, 6, 112].

### 3.6.6 Processeur virtuel

Un programme distribué envoie beaucoup de messages. Le transport de messages est beaucoup plus long qu'un accès à la mémoire centrale, et donc il est important de minimiser le nombre de messages.

Entreposer toutes les données du problème à un seul endroit impliquerait qu'aucun message n'est nécessaire. Mais ce n'est pas désirable, car un algorithme distribué doit distribuer les données et les calculs. La distribution des données peut être disjointe ou non, c'est-à-dire qu'un objet primordial peut ou non exister en plusieurs exemplaires afin de créer de la redondance ou bien encore une mémoire cache.

Une astuce populaire pour donner du travail à l'unité centrale de traitement (le processeur) pendant qu'un message est en train d'être transporté est d'entrelacer la communication et le calcul. Pour ce faire, il est nécessaire de construire de plus longs messages si ceux-ci sont initialement trop petits (à cause des dépendances entre les données). Le processeur virtuel dans RayPlatform peut exécuter jusqu'à 32 768 travailleurs. Ce nombre de travailleurs peut être modifié si nécessaire. Chacun de ces travailleurs a son propre contexte d'exécution et peut être dans un des trois états suivants à un point

dans le temps : complété, endormi, disponible. Un travailleur est complété s'il n'a plus rien à faire. Il est endormi s'il a poussé un message sur le communicateur et n'a pas encore reçu de réponse. Il est disponible s'il est prêt à exécuter son code.

Le sous-système conjoint du processeur virtuel (classe C++ VirtualProcessor) est le communicateur virtuel (VirtualCommunicator, voir la section 3.6.7). Les travailleurs qui implémentent l'interface Worker envoient des messages en utilisant l'interface de programmation de la classe *VirtualCommunicator*. Un calcul typique est fait sur plusieurs rangs MPI. Avec RayPlatform, chaque rang a plusieurs travailleurs (par défaut, le nombre est 32 768). Ces travailleurs envoient des messages au communicateur de la classe *VirtualCommunicator* et s'exécutent à l'intérieur du processeur virtuel *VirtualProcessor*. Lorsqu'un nombre suffisant de messages en provenance des travailleurs est atteint, un gros message contenant plusieurs petits messages est envoyé au département du transport pour être envoyé à une destination. Ce processus d'agglomération permet de sauver beaucoup de temps, car beaucoup d'aller-retours sont ainsi évités.

### 3.6.7 Communicateur virtuel

Le communicateur virtuel reçoit des messages des travailleurs, et doit agglomérer ces marchandises en de plus gros morceaux afin de minimiser le nombre de messages réels. Le communicateur virtuel oblige ses clients à utiliser toujours la même taille de messages unitaires afin de faciliter les événements de multiplexage et de démultiplexage. Le communicateur virtuel peut aussi forcer l'envoi d'un message dans lequel plusieurs petits messages de travailleurs sont présents si tous les travailleurs du rang sont endormis. Lorsque la réponse agglomérée à un message aggloméré est reçu, le processeur virtuel se charge de réveiller les travailleurs endormis qui ont reçu une réponse dans la réponse agglomérée.

Le communicateur virtuel est donc une composante qui permet d'agglomérer les petits messages des travailleurs et d'envoyer moins de messages (qui sont plus gros) au travers de la couche de l'interface de passage de messages. Chaque rang possède un communicateur virtuel et il n'y a pas de synchronisation entre les différents communicateurs virtuels – ceux-ci sont autonomes.

### 3.6.8 Routeur virtuel de messages

Parfois, la taille d'un jeu de données peut nécessiter un très grand nombre de rangs. Beaucoup de réseaux sont très mauvais lorsque beaucoup de rangs communiquent entre

eux en utilisant une approche n'importe qui à n'importe qui. Puisque c'est l'approche utilisée par Ray, le routage virtuel de messages est souvent nécessaire.

Dans cette section, la surface (sommets et arêtes) d'un polytope est utilisée [23, 22, 24]. Un polytope est une sorte d'objet géométrique avec un nombre de dimensions arbitraire. Chaque sommet est un point de $n$ dimensions dans la base $B$. Deux sommets sont liés par une arête si seulement une des dimensions est différente.

En utilisant la base 64 et 4096 rangs, chacun des rangs peut être représenté par deux nombres de 0 à 63. Sans routage virtuel, n'importe quel rang peut communiquer directement avec n'importe quel autre rang. Par exemple, le rang $(1, 9)$ peut envoyer directement à $(30, 51)$ en utilisant le chemin

- $((1, 9), (30, 51))$.

Il est possible de connecter ensemble 4096 rangs avec la surface d'un polytope. Dans ce contexte, le rang $(1, 9)$ doit utiliser un chemin parmi les suivants pour se rendre à $30, 51$ :

- $((1, 9), (30, 9), (30, 51))$ ;
- $((1, 9), (1, 51), (30, 51))$.

Dans la surface d'un polytope, deux sommets sont connectés si une seule des dimensions est différente. Avec la base 64 et un diamètre de 2, le degré est donc de 126. Sans le polytope, le graphe de communication contient 16 777 216 arcs. Avec le polytope, il contient 516 096 arcs, soit environ 3% du nombre initial.

Dans les systèmes informatiques distribués, les processus communiquent ensemble en s'envoyant des messages. Par défaut, le graphe complet de communication est utilisé. Dans un tel graphe, n'importe quel processus peut envoyer un message directement à n'importe quel autre processus. Pour 64 processus, chaque processus a 64 voisins (incluant lui-même). Pour 1024 processus, le nombre de voisins est 1024 et le nombre de liens est $(1024)^2 = 1048576$. Le matériel de certains superordinateurs n'est pas très bon lorsqu'il y a trop de voisins parce que seulement un nombre restreint de voisins peut résider sur le matériel. Donc, dans certains cas, un nombre trop élevé de voisins causera une latence plus élevée que la latence considérée normale pour un système donné.

Une façon de régler ce problème est l'utilisation du routage de messages. Avec le routage de messages, un processus a seulement quelques voisins directs avec lesquels il peut communiquer directement, sans intermédiaire. Les messages utilisent alors des routes qui touchent seulement aux arêtes permises. Ici, le routage avec le tore, l'hypercube et le polytope est présenté.

### Tore

Les superordinateurs Cray [270, 82] et les superordinateurs IBM Blue Gene [10, 51] utilisent un grillage pour router les messages physiquement entre les cœurs. Cette réseautique suit une topologie en forme de tore. Par exemple, le IBM Blue Gene/Q utilise un tore à 5 dimensions : A, B, C, D, E (il y a 6 dimensions si la dimension T est considérée).

Les dimensions A à E sont entre les nœuds alors que la dimension T est à l'intérieur d'un nœud (T pour *Thread*). Pour un tore de géométrie 16x16x12x12x2, lorsqu'un message est transmis de $(2, 3, 5, 1, 1)$ à $(4, 6, 2, 1, 0)$, seulement une des 5 dimensions peut changer à chaque saut. Avec l'unité de messages (*Message Unit*) du IBM Blue Gene/Q, les 10 opérations possibles sont A-, A+, B-, B+, C-, C+, D-, D+, E-, et E+. Par exemple, A- signifie qu'il faut soustraire 1 à la dimension A du tuple $(A, B, C, D, E)$. Les sauts peuvent aussi être ordonnés par direction. Pour se rendre de $(2, 3, 5, 1, 1)$ à $(4, 6, 2, 1, 0)$, la séquence d'opérations suivante peut être utilisée : A+, A+, B+, B+, B+, C-, C-, C-, E-. Ces opérations peuvent être ré-ordonnées afin d'uniformiser la charge de chaque sommet du tore.

Dans un tore, les dimensions sont en boucle. Par exemple, si la dimension A peut prendre les valeurs $0, 1, \ldots, n$ alors faire A- sur 0 est possible et donne $n$. Inversement, faire A+ sur n est aussi possible et donne 0.

Un tore peut aussi être défini avec une base différente pour chaque dimension, comme pour le tore du IBM Blue Gene/Q.

### Hypercube

Un hypercube est une figure géométrique comme le point, la ligne, le carré et le cube. Le carré est l'hypercube en 2 dimensions. Le cube est l'hypercube en 3 dimensions – ces faces sont des objets de la dimension précédente (les carrés). Les faces de l'hypercube en 4 dimensions sont des cubes. Un hypercube est défini par un nombre de dimensions $D$, et une base $B$. La base $B$ des hypercubes est toujours 2. Chaque sommet d'un

hypercube est un tuple contenant $D$ éléments entre 0 et $B-1$. Une arête existe entre deux sommets si et seulement si une des dimensions diffère d'une unité. Par exemple, pour un hypercube de 3 dimensions et de base 2, le sommet $(0,0,0)$ est lié aux sommets $(1,0,0)$, $(0,1,0)$, et $(0,0,1)$.

Dans l'hypercube et le tore, il est seulement possible de faire des sauts d'une unité. Par exemple, pour la base $B = 0,1,2,3$, il est impossible de passer de 2 à 0 en une opération pour un tore. Avec l'hypercube, puisque la base est toujours 2, alors les sauts sont toujours d'une unité nécessairement.

#### Polytope

La surface d'un polytope régulier convexe est un graphe (sommets et arêtes). Ce graphe est régulier et peut être utilisé pour router des messages dans un système complexe [23, 22, 24]. Avec l'hypercube et le tore, il y a une arête entre deux sommets si et seulement si une seule des dimensions diffère d'une unité. Pour le polytope, il y a une arête entre deux sommets si et seulement si une seule des dimensions diffère. Il n'y a pas de contrainte sur la valeur de la différence (la différence absolue doit être supérieure à 0).

Le routage par polytope est plus appréciable, car des degrés plus grands (et des diamètres plus petits) sont possibles. De plus, de par sa nature, il existe un nombre factoriel de routes de longueur minimale. Le nombre de sauts pour se rendre d'un sommet à un autre est borné par la dimension du polytope. Pour se rendre de $(2,3,5,1,1)$ à $(4,6,2,1,0)$, il faut appliquer au plus 5 vecteurs. Dans cet exemple, les 5 vecteurs sont : $(2,0,0,0,0)$, $(0,3,0,0,0)$, $(0,0,-3,0,0)$, $(0,0,0,0,0)$ et $(0,0,0,0,-1)$. Seulement les 4 vecteurs qui ont une norme non nulle sont pertinents. Les 4 vecteurs de saut peuvent être appliqués dans n'importe quel ordre, ce qui permet de balancer le graphe de communication.

### 3.6.9 Mini rangs

Les architectures matérielles sont hiérarchiques en débutant par la douille (*socket*) du processeur, le processeur, les cœurs, et finalement les fils matériels (*threads*). Cependant, le standard informatique appelé MPI (Message Passing Interface) prévoit seulement des processus qui s'échangent des messages – une architecture à plat sans aucun niveau. Les minis rangs sont un modèle hybride qui unit les processus MPI avec les fils d'exécution.

Ce modèle a été imaginé par moi-même, Fangfang Xia et Rick Stevens. Ce modèle est

similaire aux acteurs [112, 5], mais moins puissant, car il est moins formel. Ce modèle consiste à exposer uniquement une boîte de réception de messages et une boîte d'envoi de messages à chaque instance de l'application logicielle. Ces instances, appelées minis rangs, s'exécutent dans des fils d'exécution *IEEE POSIX thread*, qui sont eux-mêmes à l'intérieur d'un processus qui correspond à un rang MPI. Tous les messages vont suivre des routes avec des sauts, car un mini rang ne peut pas communiquer directement avec un autre mini rang. Pour ce faire, un mini rang dépose son message dans sa boîte de sortie. Ensuite, RayPlatform transfère ce message dans la boîte non bloquante partagée du mini rang pour rendre le message disponible au fil d'exécution du processus qui est en charge de faire les appels de fonctions dans la bibliothèque logicielle MPI. L'approche MPI+MPI est très similaire et permet de partager des segments de mémoire entre les rangs MPI qui résident sur le même nœud de calcul [115]. Bref, les architectures hybrides qui utilisent le passage de messages et les fils d'exécution sont supérieures, car elles s'adaptent mieux aux architectures hiérarchiques existantes.

### Queue de communication non bloquante

Afin d'utiliser les fils d'exécution, il est requis d'avoir une méthode de synchronisation en temps réel à faible contention. Idéalement, aucun cadenas ne doit être présent. Sur les ordinateurs, la lecture ou la mise à jour d'un mot de mémoire est atomique, car la largeur du bus système est par définition égale à la taille d'un mot de mémoire. Il est donc possible d'exploiter ce fait pour créer une queue de communication à 1 producteur et à 1 consommateur qui est non bloquante. Pour ce faire, un anneau de $N + 1$ cellules est créé où $N$ est le nombre maximal de messages qui peuvent exister en tout temps dans l'anneau. Deux pointeurs de cellule sont requis : la tête où se fait les tractions par le consommateur et la queue où se fait les poussées par le producteur. Si la tête et la queue sont sur la même cellule, alors il n'y a strictement aucun message. Si la queue est avant la tête, il y a exactement $N$ messages. Dans tous les autres cas, il y a nécessairement au moins 1 message et au plus $N - 1$ messages. Ce genre de modèle est utilisé de manière beaucoup plus sophistiqué dans le noyau Linux sous le modèle RCU (*Release-Copy-Update*).

# Chapitre 4

# Graphe de de Bruijn et assemblage *de novo*

J'ai utilisé le graphe de de Bruijn comme concept fondamental pendant mon doctorat afin de modéliser et de structurer toute l'information fournie par le séquençage à haut débit.

Le graphe de de Bruijn permet de représenter de manière très compacte l'ensemble des mots génétiques présents dans un génome donné, en utilisant les mots génétiques présents dans les lectures de ce même génome (le génome est non-disponible, mais les lectures sont disponibles) [217, 219, 307]. Cette astuce peut servir de base pour l'assemblage *de novo* de génomes. Mais en premier, il est nécessaire de définir le concept de graphes. Les graphes sont définis dans la prochaine section.

## 4.1 Graphes

Un *graphe (orienté)* $G$ est une représentation d'une relation entre des paires d'éléments d'un ensemble de points que l'on nomme sommets. L'existence d'une relation (ou lien) entre deux sommets $u$ et $v$ est représenté par un arc $(u, v)$. On dénote donc un graphe $G = (V, E)$ où $V$ est l'ensemble des sommets et $E \subseteq V \times V$ est l'ensemble des arcs. les arêtes sont $E \subseteq V \times V$. Donc, les sommets sont représentés par l'ensemble $V$ et les arcs (arêtes dirigées) sont représentés par l'ensemble $E$. Les arêtes représentent des relations entre les éléments de l'ensemble $V$. Les relations peuvent être ou non symétriques, c'est-à-dire que $(u, v) \in E \Leftrightarrow (v, u) \in E$. Dans le cas où la relation est symétrique, on représente les deux arcs $(u, v)$ et $(v, u)$ par $[u, v]$. On appelle cette dernière une *arête* et on parle dans ce cas de graphe *non-orienté*. Les sommets et les arêtes du carré, un

polytope, forment le graphe $G = (V, E)$ où
$V = \{(0,0), (0,1), (1,0), (1,1)\}$ et où
$E = \{[(0,0), (0,1)], [(0,0), (1,0)], [(1,1), (0,1)], [(1,1), (1,0)]\}$. Les graphes sont utilisés par beaucoup de sites web. Sur Facebook, les sommets sont les personnes et les arêtes sont les relations d'amitié. Google s'intéresse aux pages web – les sommets sont les pages web et les arêtes sont les hyperliens entre ces pages web. L'algorithme de Google appelé PageRank utilise un énorme graphe pour calculer quelles pages web il est pertinent de montrer à un utilisateur suivant une recherche par mots-clés. Le réseau Internet contient des routeurs et des liens.

Un *graphe complet* sur un ensemble de sommets $V$ est un graphe où il y a une arête entre les deux sommets de chaque paire de sommets distincts, c'est-à-dire que $\forall u, v \in V, u \neq v, \ [u, v] \in E$. Un *sous-graphe* est un graphe qui contient un sous-ensemble des sommets et/ou un sous-ensemble des arêtes d'un autre graphe.

Le *degré* d'un sommet est son nombre d'arêtes. Si le graphe est orienté, une distinction est faite entre le degré entrant et le degré sortant d'un sommet en fonction du nombre d'arcs entrants et sortants Un graphe est dit *régulier* si tous les sommets ont le même degré.

Un *chemin* dans un graphe est une suite de sommets $\langle v_1, v_2, v_3, \ldots, v_{n-2}, v_{n-1}, v_n \rangle$ tel que $\forall i \in \{1, 2, \ldots, n-1\} \ \ (v_i, v_{i+1}) \in E$. La notion de chemin sera utilisée plus tard dans ce document pour l'assemblage *de novo* de génomes en utilisant un graphe de de Bruijn (voir plus bas). Un exemple pratique de chemin est la route de livraison utilisée lorsqu'une lettre est envoyée par la poste. Un autre exemple similaire est la route utilisée par la transmission d'un message électronique. Finalement, afin de donner un exemple réel, la figure 4.1 contient une route sur le graphe du réseau Internet entre le Centre de recherche du CHU de Québec et l'Université de Tokyo.

```
$ tracepath www.u-tokyo.ac.jp
 1:  192.168.128.188                                       0.119ms pmtu 1500
 1:  192.168.128.1                                         0.367ms
 1:  192.168.128.1                                         0.338ms
 2:  132.203.117.1                                         0.612ms
 3:  gw1112.n.ulaval.ca                                    1.574ms
 4:  132.203.253.56                                        1.291ms
 5:  dknbin-g0-2.n.ulaval.ca                               1.045ms
 6:  cslutmbiex.n.ulaval.ca                                1.015ms
 7:  gw541a.n.ulaval.ca                                    1.273ms
 8:  ulaval-gw.risq.net                                    1.462ms
 9:  ulaval-internet-dqubc-ul.risq.net                     1.735ms
10:  imtrl-rq.risq.net                                     8.477ms asymm 12
11:  te-4-1.car2.montreal2.level3.net                     14.268ms asymm 20
12:  ae-10-10.ebr1.chicago1.level3.net                    84.775ms asymm 21
13:  ae-6-6.ebr1.chicago2.level3.net                      84.992ms asymm 21
14:  ae-3-3.ebr2.denver1.level3.net                       86.202ms asymm 21
15:  ae-1-100.ebr1.denver1.level3.net                     85.307ms asymm 21
16:  ae-3-3.ebr2.sanjose1.level3.net                      85.071ms asymm 21
17:  ae-82-82.csw3.sanjose1.level3.net                    91.123ms asymm 20
18:  ae-1-60.edge1.sanjose2.level3.net                    85.026ms asymm 19
19:  kddi-americ.edge1.sanjose2.level3.net               179.912ms
20:  pajbb001.int-gw.kddi.ne.jp                           85.895ms asymm 19
21:  otejbb206.int-gw.kddi.ne.jp                         189.837ms asymm 19
22:  jc-ote301.int-gw.kddi.ne.jp                         189.919ms asymm 20
23:  124.211.10.42                                       200.062ms asymm 24
24:  no reply
25:  no reply
26:  no reply
27:  59.106.9.254                                        200.565ms
28:  ip-59-106-161-2.adm.u-tokyo.ac.jp                   173.624ms asymm 26
29:  59.106.161.28                                       196.518ms !H
     Resume: pmtu 1500
```

FIGURE 4.1: Exemple de route entre le Centre de recherche du CHU de Québec et Todai (L'Université de Tokyo).

Le nombre dans la première colonne est le numéro du saut. La deuxième colonne contient le nom du nœud sur Internet. Finalement, la troisième colonne contient le temps écoulé depuis le début pour atteindre cet endroit.

Un chemin *eulérien* est un chemin dans un graphe qui utilise toutes les arêtes une et une seule fois. L'approche algorithmique proposée par Pevzner et ses collaborateurs [219] utilise une telle méthode pour résoudre le problème de l'assemblage *de novo* en cherchant un chemin eulérien dans un certain graphe (le graphe de de Bruijn, définit plus bas dans la section 4.2). Cette approche ne fonctionne pas en présence d'erreurs de séquençage.

## 4.2 Origine du graphe de de Bruijn

Nicolaas Govert de Bruijn a introduit le graphe de de Bruijn en 1946 [66]. Il a démontré que le graphe de de Bruijn peut être utilisé pour générer des séquences de de Bruijn.

Sur un alphabet $\Sigma$ donné, un *mot cyclique* est un mot dont les lettres sont écrites sur les perles d'un collier. Ainsi, par exemple, on ne peut pas distinguer les mots abc, cab et bca, ils correspondent tous au même mot cyclique.

Une *séquence de de Bruijn* est un mot cyclique sur l'alphabet $\Sigma$ qui contient tous les mots de longueur $n$ exactement une seule fois un mot cyclique sur un alphabet. L'exemple donné dans l'article [66] est le mot cyclique de type $P_3$ $\overrightarrow{00010111}$. Ce mot cyclique contient tous les mots de longueur 3 possibles avec l'alphabet $\{0, 1\}$ : 000, 001, 010, 101, 011, 111, 110, 100. Notez que pour retrouver les mots 110 et 100, il faut respectivement considérer faire deux ou une rotation(s) du mot $P_3$ afin d'obtenir 11000101 et 10001011. Dans l'article [66], des formules sont données pour dénombrer des chemins qui ont certaines propriétés dans le graphe de de Bruijn.

Le graphe de de Bruijn est défini à partir d'un entier $k \in \mathbb{N}$ ainsi qu'un alphabet fini $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_{|\Sigma|}\}$. Le graphe $G_{dedeBruijn}(\Sigma, k) = (V, E)$ possède les sommets $V = \Sigma^k$ et les arêtes dirigés $E \subseteq V \times V$.

Comme un $k$-tuple $(\tau_1, \tau_2, \ldots, \tau_k) \in \Sigma^k$ peut être vu comme le mot $\tau_1\tau_2..\tau_k$, les sommets du graphe de de Bruijn sont donc tous les mots de longueur $k$ possibles à partir de l'alphabet $\Sigma$.

Les arcs du graphe de de Bruijn sont définis par

$$(u_1u_2 \ldots u_k, v_1v_2 \ldots v_k) \in E \Leftrightarrow u_i = v_{i-1} \forall i \in \{2, 3, \ldots, k-3, k-2, k-1, k\}. \quad (4.1)$$

Le nombre de sommets est donc $|V| = |\Sigma|^k$. Par exemple, avec $\Sigma = \{0, 1\}$ et $k = 64$, le

graphe de de Bruijn contient $2^{64} = 18\ 446\ 744\ 073\ 709\ 551\ 616$ sommets. La relation suivante définit les arêtes dirigées :

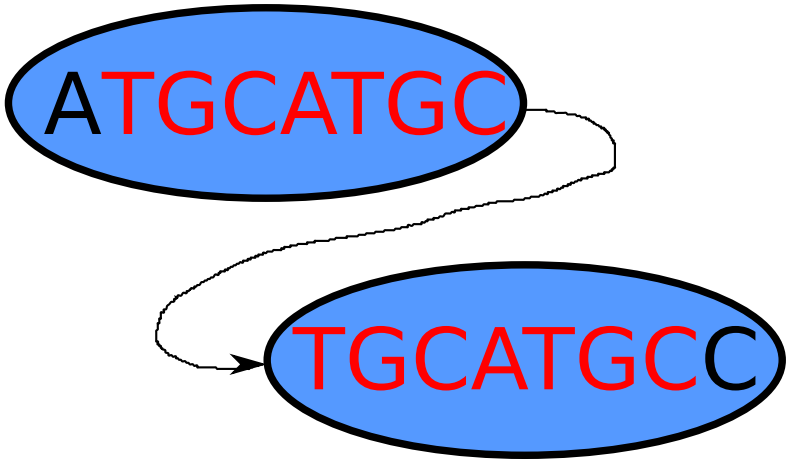


FIGURE 4.2: Relation de de Bruijn.

La relation de de Bruijn entre deux séquences d'ADN est montrée en rouge.

Donc, chaque sommet du graphe a exactement $|\Sigma|$ parents et $|\Sigma|$ enfants. Pour générer l'ensemble des enfants d'un sommet quelconque $u = \langle u_1, u_2, u_3, u_4, \ldots, u_{k-2}, u_{k-1}, u_k \rangle$, il faut prendre sa séquence de symboles, les bouger tous une fois à gauche, puis créer un enfant pour chaque symbole de l'alphabet en l'ajoutant à la fin. La figure 4.2 illustre la relation de de Bruijn entre deux séquences d'ADN.

Si pour chaque arc $(u, v)$, on dit que $v$ est un enfant de $u$ et que $u$ est un parent de $v$, alors pour obtenir l'ensemble des enfants d'un sommet $u \in V$, il faut faire

$$enfants(u) = \bigcup_{i=1}^{|\Sigma|} \{\langle u_2, u_3, u_4, \ldots, u_{k-2}, u_{k-1}, u_k, \sigma_i \rangle\}. \tag{4.2}$$

De manière similaire, les parents d'un sommet sont

$$parents(u) = \bigcup_{i=1}^{|\Sigma|} \{\langle \sigma_i u_1, u_2, u_3, u_4, \ldots, u_{k-2}, u_{k-1} \rangle\}. \tag{4.3}$$

Le graphe de Bruijn peut être utiliser comme topologie pour faire du routage (voir la section 3.6.8). Avec un graphe de de Bruijn (en fait, avec un sous-graphe du graphe de de Bruijn), on peut représenter un échantillon de lectures obtenues par un ou plusieurs séquenceurs. Entre autres, une lecture d'ADN se représente par un chemin dans le

graphe et une séquence contigüe n'est qu'un certain chemin (plus long) dans ce même graphe. Les sections 4.7.3 et 4.7.4 contiennent plus de détails sur ces applications du graphe de de Bruijn.

Le degré entrant du graphe de de Bruijn est régulier et sa valeur est de $|\Sigma|$ tandis que le degré sortant du graphe de de Bruijn est aussi régulier et est aussi de $|\Sigma|$. Le diamètre du graphe de de Bruijn est toujours $k$, c'est-à-dire que pour n'importe quelle paire de sommets $(u, v) \in V \times V$, il existe un chemin qui contient au plus $k$ sommets dans le graphe de de Bruijn $P_{u,v} = cheminLePlusCourt(u, v) = \langle p_1 = u, p_2, p_3, \ldots, p_{n-2}, p_{n-1}, p_n = v \rangle$ où $n \leq k$. Le graphe de de Bruijn a donc des propriétés intéressantes car il est régulier et contient relativement beaucoup de sommets étant donné son faible diamètre [285]. En résumé, le graphe de de Bruijn est défini à partir d'un alphabet $\Sigma$ et d'un entier $k \in \mathbb{N}$. En raison de son faible diamètre, le graphe de de Bruijn peut sembler être un bon candidat pour faire du routage de messages (le routage est décrit dans la section 3.6.8). Malheureusement, comme entre deux sommets il n'existe qu'un seul chemin le plus court, certains arcs du graphe de de Bruijn se retrouvent dans un grand nombre de ces chemins, créant ainsi des embouteillages lorsqu'une grande quantité de paires de sommets du graphe veulent s'envoyer des messages en même temps. Alors, le graphe de de Bruijn n'est pas vraiment un bon graphe pour faire du routage de messages puisqu'il existe un seul chemin le plus court entre deux sommets et donc il n'est pas possible de balancer les messages sur d'autres chemins de la même longueu puisqu'il y en a juste un de longueur minimale. De plus, la distribution du nombre de routes passant par chacun des sommets n'est pas uniforme.

## 4.3 Graphe de Kautz

Le graphe de Kautz est un sous-graphe du graphe de de Bruijn [285]. Pour l'obtenir, il suffit d'enlever tous les sommets qui contiennent au moins deux symboles consécutifs identiques. Tous les sommets qui contiennent deux symboles consécutifs identiques ne sont pas admissibles dans le graphe de Kautz. Par exemple, le sommet 001010101 est un sommet du graphe de de Bruijn. Ce n'est pas le cas dans le graphe de Kautz, car 001010101 contient la séquence 00, et que 0 et 0 sont consécutifs et identiques. Le graphe de Kautz a le plus petit diamètre possible pour un nombre de sommets donné. Donc, pour un degré et un nombre de sommets donnés, le graphe de Kautz a le diamètre le plus bas mathématiquement. Cette propriété fait donc du graphe de Kautz un candidat idéal pour construire des systèmes avec un nombre très petit de liens.

Tout comme le graphe de de Bruijn, le graphe de Kautz ne permet pas de faire du routage adaptatif avec un ensemble de chemins les plus courts puisqu'il existe un seul chemin le plus court entre une paire de sommets. Dans ce cas-ci, on aura également des problèmes d'embouteillage. La compagnie SiCortex – une compagnie qui n'existe plus – utilisait le graphe de Kautz pour construire la réseautique de ses superordinateurs. Les superordinateurs de SiCortex utilisaient très peu d'énergie, et le transport des messages était réalisé avec une topologie en graphe de Kautz. Avec la base $r$ et des sommets de longueur $k$, le graphe de Kautz a $(r) * (r-1)^{(k-1)}$ sommets car il est possible de choisir parmi les $r$ symboles pour la première position alors que les autres positions ne peuvent pas réutiliser le symbole présent à la position précédente. En pratique, ce graphe n'est pas beaucoup utilisé, car le nombre de sommets d'un graphe de Kautz est difficile à obtenir. En effet, en pratique les superordinateurs utilisent des multiples de puissances de 2.

## 4.4 Topologies caractéristiques

Dans le contexte de l'assemblage *de novo*, Zerbino et al. [307] a introduit les impasses (*tips*) et les bulles. Ces concepts avaient aussi été introduit avant par Pevzner *et al.* [219]. Ce sont des structures présentes dans un graphe de de Bruijn. Une bulle est formée de deux branches et ces deux branches se rejoignent aux deux extrémités. La figure 4.3 montre une bulle et des impasses.

Les bulles sont produites lorsqu'un site est hétérozygote ou bien lorsqu'une erreur de séquençage est présente à un site. Dans le cas ou une position dans le génome est hétérozygote, l'amplitude du signal pour chacune des allèles est en général très similaire. Lorsqu'une telle bulle est causée par des erreurs de séquençage, la profondeur de séquençage pour la vraie valeur est élevée alors que la profondeur de séquençage associée à l'erreur est beaucoup plus basse. Les impasses, quant à eux, contiennent une seule branche et seulement une des extrémités est liée au reste du graphe.

## 4.5 Comparaison d'échantillons

La topologie s'intéresse à l'agencement des éléments entre eux. La topologie peut être utilisée pour analyser le graphe de de Bruijn d'un échantillon de lectures obtenues par séquençage de l'ADN (section 4.7.3 ). Par exemple, les bulles qui ne sont pas causées par des erreurs de séquençage sont en général produites par des variations génétiques.

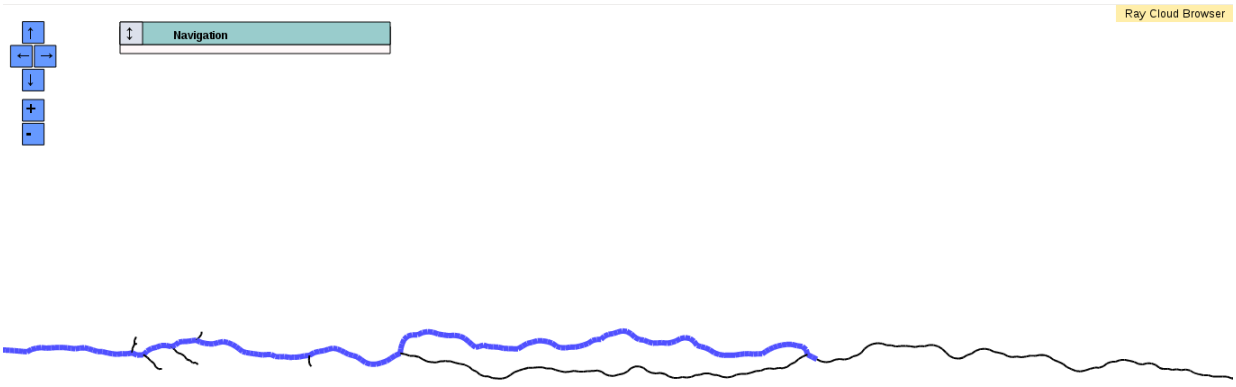


FIGURE 4.3: Une bulle et cinq impasses.
Au centre, on peut voir une bulle avec deux chemins qui commencent et terminent au même endroit. À gauche de cette bulle, on peut voir cinq impasses (les petites lignes noires qui dépassent du chemin bleu).

Cette observation est la base des concepts de graphes coloriés introduits par Iqbal et ses collaborateurs [126]. Dans un tel graphe colorié, on note la provenance du signal sur les sommets du graphe de façon à pouvoir voir, par exemple, les sommets qui sont spéficiques à un échantillon. Un graphe colorié est aussi appelé un graphe étiqueté.

Ce même graphe peut être annoté avec un vocabulaire connu (en général, les mots d'ADN appartenant à des génomes ou à des gènes connus) afin de construire des rapports d'abondances.

## 4.6 Assemblage de génomes

L'assemblage *de novo* de génomes est le processus par lequel une grande quantité de morceaux d'ADN – lesquels sont appelés les lectures – sont mis ensemble afin de reconstruire des morceaux de plus grande taille [229, 198]. Ce type d'analyse nécessite des graphes, comme un graphe de de Bruijn, pour stocker et travailler sur les données. Mon projet principal de doctorat était Ray. Ray est un assembleur *de novo* de génomes [30], de méta-génomes [32] et de transcriptomes. L'article décrivant les heu-

ristiques et les concepts utilisés dans l'architecture Ray a été publié dans le *Journal of Computational Biology* [30]. Ray est une pile de modules d'extension compatible avec l'engin RayPlatform (voir le chapitre 3). L'assemblage *de novo* est donc un autre type d'analyse des séquences [212], un type d'analyse différent des alignements. Essentiellement, l'assemblage consiste en la construction de longues séquences à partir de plus courtes séquences. Différents algorithmes et structures de données existent pour cette tâche. Les deux principaux graphes sont le graphe à chevauchements et le graphe de de Bruijn. Pour une revue de la littérature, voir [190, 229] Certains assembleurs peuvent aussi calculer des variations génétiques [126, 157]. Dans la section suivante, le lien entre l'assemblage de génomes et le graphe de de Bruijn est présenté.

## 4.7 Lien avec l'assemblage de génomes

### 4.7.1 Assembler sans graphe

Des assemblages de génomes peuvent être obtenus sans utiliser de graphe de de Bruijn ou d'autres types de graphes. Plusieurs assembleurs simples qui n'utilisent pas de graphe ont été proposés : SSAKE [296], SHARCGS [72], VCAKE [130], et QSRA [38]. L'idée principale de ces outils est d'utiliser un algorithme vorace par extension. Le choix de la prochaine lecture à ajouter est fait en utilisant uniquement l'information locale. Ces méthodes n'utilisent pas de graphes.

L'histoire racontée ici énumère les étapes importantes qui ont mené jusqu'à l'utilisation du graphe de de Bruijn en génomique. Comme l'indiquent les paragraphes suivants, le graphe de de Bruijn a eu plusieurs prédécesseurs au cours des années.

### 4.7.2 Graphe à chevauchements

Les sommets du graphe à chevauchements sont les lectures d'ADN et les arêtes sont les chevauchements entre ces lectures d'ADN. Le premier article dans la littérature décrivant le graphe à chevauchement a été publié en 1983 par Gallant [92]. Cet article a évalué la complexité de l'assemblage de fragments avec des graphes. Par la suite, le premier génome assemblé avec un logiciel a été rapporté en 1995 [84] ; les auteurs y ont décrit le génome de la bactérie *Haemophilus influenzae Rd.* L'article classique décrivant l'assemblage de *Drosophila melanogaster* [195] – une espèce de mouche – a popularisé la méthode d'assemblage avec un graphe à chevauchements à partir de données de séquençage de type *shotgun*. Dans ce même article, l'assembleur Celera a été décrit.

De plus, l'assembleur Celera a été décrit en détail comme étant très modulaire avec des compartiments pour chaque composante logicielle [121], menant ainsi à un système robuste et modulaire.

L'assembleur ARACHNE est aussi un assembleur qui utilise un graphe à chevauchements [21]. Les sommets du graphe à chevauchements sont les séquences d'ADN. Les arêtes sont les chevauchements. Les graphes à chevauchements consomment trop de mémoire (espace) lorsque le nombre de séquences est élevé. Le nombre de séquences est élevé avec le séquençage à haut débit. Beaucoup de recherche a été investie dans l'amélioration des méthodes pour obtenir des chevauchements. Un élément important est de minimiser le nombre de chevauchements dans la catégorie des faux positifs. La première formulation mathématique a été proposée par Myers [194]. Des méthodes pour éliminer les mauvais chevauchements ont aussi été proposées. Celles-ci utilisent les filtres avec des q-gram [237, 177]. Les trois assembleurs les plus populaire pour les données 454 sont wgs-assembler [68, 189], Newbler [177], et Mira [52].

### 4.7.3 Graphe de de Bruijn

En 1989, un travail clé a été publié décrivant une méthode exhaustive pour énumérer les mots d'ADN présents dans un échantillon, ainsi que l'abondance associée à chaque mot [217]. Cette méthode se basait sur la complémentarité des brins d'ADN. L'ADN complémentaire – ou inverse complément – est la séquence d'ADN (simple brin) qui permet de compléter une séquence d'ADN afin d'obtenir une molécule avec deux brins d'ADN. Dans les règles d'appariement, l'adénine va avec la thymine (et vice-versa) et la cytosine va avec la guanine (et vice-versa).

La méthode de [217] n'utilisait pas le séquençage de l'ADN, mais utilisait plutôt l'hybridation sur biopuces. Dans ce contexte, les sondes qui ont capturé du matériel dans l'échantillon indiquent que les séquences correspondantes aux sondes (ou les compléments inverses de celle-ci) sont présentes dans l'échantillon. Le problème principal de cette approche était que la longueur des sondes – et aussi leur nombre – étaient physiquement limités par la surface disponible sur la biopuce.

Un peu plus tard, Idury et Waterman ont adapté la méthode pour l'assemblage *de novo* [125]. En 2001, une méthode utilisant les chemins Eulériens dans un graphe de de Bruijn a été proposée comme une alternative au graphe à chevauchements [219]. L'avantage principal du graphe de de Bruijn est que la taille théorique du sous-graphe du graphe de de Bruijn induit par les lectures d'ADN est constante en fonction du génome d'où

proviennent ces lectures et ne dépend pas de la quantité de données.

En pratique, les erreurs de séquençage produisent des sommets supplémentaires [60], et donc le nombre de sommets n'est pas borné.

Le projet international du génome humain n'a pas utilisé de graphes de de Bruijn. En effet, les graphes à chevauchement furent utilisés par l'effort public [145, 100] et par l'effort privé de la compagnie Celera [197]. Par la suite, l'arrivée du séquençage à haut débit a donné un second souffle aux graphes de de Bruijn, lesquels avaient été introduit par Pevzner *et al.* en 2001 [219]. L'assembleur Velvet a été le premier assembleur de de Bruijn à gérer les gros jeux de données de séquençage [307]. Un article expliquant l'utilisation du graphe de de Bruijn en génomique a été publié en 2011 [58].

Aucun assembleur n'utilise le graphe de de Bruijn complet, car il contient beaucoup trop de sommets. Les sommets dans les séquences sont les sommets dans le génome [46]. Les assembleurs de de Bruijn utilisent un sous-graphe du graphe de de Bruijn et non un graphe de de Bruijn [32]. La majorité des assembleurs de de Bruijn utilisent une seule valeur ($k$) pour la longueur des sommets. IDBA est un assembleur itératif utilisant plusieurs valeurs de $k$ [216, 214, 215]. SPAdes est un autre assembleur qui fonctionne aussi lorsque les données de séquençage ont été obtenues à partir d'une seule cellule [17, 225].

### 4.7.4 Assemblage *de novo* avec un graphe de de Bruijn

Dans cette section, une définition du graphe de de Bruijn appliqué à la génomique est donnée. En biologie, le graphe de de Bruijn est utilisé pour représenter des séquences d'ADN. Le graphe de de Bruijn pour la génétique est défini avec l'alphabet $\Sigma = \{A, T, C, G\}$ et avec un entier $k \in \mathbb{N}$. Le graphe de de Bruijn $G = (V, E)$ possède les sommets $V = \Sigma^k$, c'est-à-dire toutes les séquences d'ADN de longueur $k$. La relation pour les arêtes n'est pas symétrique est définie avec $(u_1u_2 \dots u_k, v_1v_2 \dots v_k) \in E \Leftrightarrow u_i = v_{i-1} \forall i \in \{2, 3, \dots, k-3, k-2, k-1, k\}$. Donc, si deux mots d'ADN ont un chevauchement de $k-1$ nucléotides, une arête est présente entre ces deux mots. Dans le graphe de de Bruijn, un sommet est donc un mot de longueur $k$ et une arête est un mot de longueur $k+1$. Le diamètre du graphe de de Bruijn est $k$. Le degré sortant d'un graphe de de Bruijn est 4 car $|\Sigma| = 4$. Le degré entrant est aussi 4.

En génomique, le graphe de de Bruijn est utilisé comme gabarit, c'est-à-dire qu'initialement tous les sommets sont considérés comme étant absents. Le graphe de de Bruijn

contient $4^k$ sommets et $4^{(k+1)}$ arêtes. Par exemple, il y a 4 611 686 018 427 387 904 mots de longueur 31 différents avec l'alphabet $\{A, T, C, G\}$. Il est simplement impossible de considérer tous ces mots. L'astuce est d'utiliser un sous-graphe du graphe de de Bruijn.

Pour obtenir un sous-graphe du graphe de de Bruijn qui représente bien un échantillon biologique, seulement les mots d'ADN de longueur $k$ observés dans les données seront considérés pour les sommets. Autrement dit seulement les mots de longueur $k$ qui sont des sous-mots d'une des lectures d'ADN observées sont considérés. Pour ce faire, la profondeur de séquençage d'un mot particulier est incrémentée à chaque fois qu'il est rencontré. De plus, seulement les mots de longueur $k+1$ qui sont sous-mots d'une des lectures d'ADN observées seront considérés pour les arêtes. Bien sûr, il faut utiliser les deux brins de l'ADN pour inférer les sommets du sous-graphe à construire, puisque les séquences numériques d'ADN peuvent provenir de n'importe quelle des brins.

Il y a un lien déjà remarquable entre le graphe à chevauchements et le graphe de de Bruijn ; les arêtes du graphe de de Bruijn représentent des chevauchements de $k-1$ nucléotides sur $k$ entre deux mots de longueur $k$ du graphe. Étant donné un ensemble de lectures d'ADN toutes de la même longueur $l$ (en nucléotides), un graphe à chevauchements fait avec ces lectures est équivalent à un sous-graphe du graphe de de Bruijn , avec $k = l$, construit avec ces mêmes lectures (qui ont toutes la même longueur) [196]. Dans ce cas, une lecture d'ADN correspond à une arête. En effet, le fait de considérer des valeur de $k < l$, fait qu'une lecture d'ADN correspond plutôt à un chemin du graphe (plusieurs arêtes), mais la taille de ce graphe est alors beaucoup réduite, d'où l'intérêt de considérer les graphes de de Bruijn pour le problème d'assemblage de génome. Il est préférable d'avoir $k < l$ afin de générer les arêtes du sous graphe de de Bruijn (qui sont des séquences de $k+1$ nucléotides) sans devoir comparer les lectures entre elles. Dans le cas $k = l$, il faut comparer les lectures entre elles.

### 4.7.5 Graphe de de Bruijn bidirigé

Dans le graphe de de Bruijn classique, une séquence et son complément inverse sont entreposés dans deux sommets distincts. Plusieurs assembleurs enregistrent implicitement la moitié des sommets, car une séquence d'ADN peut être obtenue à partir de son complément inverse. Le graphe bidirigé consiste à représenter explicitement les deux sommets dans une seule instance. Ce type de représentation est appelé *molécule* par Medvedev *et al.* [186, 185]. Une telle structure de données est plutôt difficile à implémenter (communication personnelle de Michael Brudno), mais les algorithmes utilisant

cette structure seront simplifiés, car en naviguant un brin d'ADN, l'autre brin est automatiquement navigué aussi (explicitement). Plusieurs logiciels représentent un sommet et son complément inverse au même endroit en mémoire, sans toutefois utiliser des arêtes bidirigées.

### 4.7.6 Graphe de de Bruijn en chaîne

Un sous-graphe du graphe de de Bruijn peut contenir un très grand nombre de sommets. Un très grand nombre de sommets peut s'avérer problématique dans certaines situations. Par exemple, si les sommets sont tous contrôlés par le même processus, il peut s'avérer difficile d'en faire une gestion efficace en mémoire. Le graphe de de Bruijn en chaîne consiste à réduire le nombre de sommets dans le graphe de de Bruijn pour obtenir un nouveau graphe contenant moins de sommets avec certains sommets plus longs. L'idée générale d'une telle approche est de concaténer deux sommets $u$ et $v$ si le sommet $v$ est le seul enfant du sommet $u$ et si le sommet $u$ est le seul parent du sommet $v$. La longueur du sommet $w = u + v$ (où le + représente une concaténation) est $|u| + |v| - k$. C'est ce principe de base qui est utilisé par plusieurs méthodes d'assemblage. Cette méthode a été introduite initialement par Pevzner en 2001 [219] sous le nom de transformation équivalente. Les assembleurs EULER [46, 49, 44, 45], Velvet [307, 308, 306], ABySS [276], SOAPdenovo [172] utilisent les transformations équivalentes ou une approche très similaire. Une limitation de cette approche est que la concaténation de deux sommets mène à une perte d'information, car les profondeurs de séquençage des sommets $u$ et $v$ doivent, en toute généralité, être combinées pour assigner une profondeur de séquençage au nouveau sommet $w$. Une façon de procéder est de calculer une moyenne pondérée en utilisant la longueur du sommet $u$ et du sommet $v$. La formule permettant de calculer, pour deux sommets $u$ et $v$, la profondeur de séquençage résultante de la concaténation des sommets $u$ et $v$ – une opération qui génère un nouveau sommet $w$ – est :

$$w.profondeur = (|u| * u.profondeur + |v| * v.profondeur)/(|u| + |v|). \quad (4.4)$$

L'assembleur ALLPATHS utilise également une formulation modifiée du graphe de de Bruijn [39, 96]. Il est à noter qu'il existe un processus similaire applicable au graphe à chevauchements dans l'optique de diminuer l'espace utilisé par la structure. Dans la section suivante, le graphe à chaîne – une généralisation du processus de simplification d'un graphe de de Bruijn ou d'un graphe à chevauchements – est introduit.

### 4.7.7 Graphe à chaînes

Jusqu'ici, les deux types de graphe important pour l'assemblage ont été présentés : le graphe de de Bruijn et le graphe à chevauchements. Le graphe de de Bruijn contient des sommets tous de la même longueur – $k$ nucléotides pour être exact – et les arêtes sont définies par la relation de de Bruijn ; cette relation est essentiellement un chevauchement de $k - 1$ nucléotides. Le graphe à chevauchements, comme son nom l'indique, contient des chevauchements. Une unification de ces deux graphes s'appelle le graphe à chaînes (*string graph*) [196, 186]. Cette formulation est le fruit de plusieurs années de recherche par Eugene Myers. Simpson et Durbin ont récemment donné une formulation du graphe à chaînes dans un cadre de compression et a aussi distribué une implémentation de référence appelée SGA [275].

## 4.8 Engin de stockage distribué

### 4.8.1 Construction d'un graphe distribué

Afin d'assembler des régions génomiques, il est nécessaire de transformer les données brutes en un format avec lequel il est envisageable de faire des calculs. La structure de données utilisée ici est le graphe de de Bruijn. Un tel graphe peut être distribué par un système de calcul en utilisant des fonctions de hachage [139].

### 4.8.2 Préparation du graphe

Pour un entier $k$ et l'alphabet $\Sigma = \{A, T, C, G\}$, les sommets du graphe de de Bruijn $G_{deBruijn}(k, \Sigma)$ sont tous les mots de longueur $k$ possibles avec l'alphabet $\Sigma$. Nous appelons ces mots des *k-mers*. À partir de données de séquençage, Ray prépare un sous-graphe du graphe de de Bruijn en observant les mots de longueur $k$ dans ces données pour définir les sommets et en observant les mots de longueur $k + 1$ pour les arêtes. Chaque rang MPI (voir le chapitre 3) utilise un tampon (un tableau d'octets pour entreposer et accumuler de l'information) pour chacun de ses destinataires. Les mots calculés à partir des séquences d'ADN (les lectures) sont déposés dans les tampons de destinataire en utilisant une fonction de hachage. Lorsqu'un tampon est plein après y avoir déposé un mot, ce tampon est envoyé à son destinataire et le rang prend une note qui lui indique qu'il doit attendre une réponse en provenance du destinataire avant de poursuivre le calcul des mots à partir des séquences. Le sous-graphe du graphe est prêt

lorsque chaque rang a terminé de calculer les mots à partir des lectures qu'il possède et qu'il n'est pas en attente d'une réponse.

### 4.8.3 Élimination précoce des erreurs

De manière générale, une erreur à la position $i$ dans une séquence contamine les mots de longueur $k$ débutant aux positions allant de $i-k+1$ jusqu'à $i$. Une erreur de séquençage (une substitution) génère exactement $k$ nouveaux $k$-mers, lesquels correspondent à une séquence d'ADN de $2k-1$ nucléotides. Il est donc important de pouvoir les détecter afin de diminuer l'utilisation de la mémoire des ordinateurs. Pour ce faire, il est possible d'utiliser un filtre de Bloom [213, 258].

Un filtre de Bloom fait ses opérations sur un ensemble d'objets $X$. Il contient $M$ bits qui sont initialement mis à 0, numérotés de 0 à $M-1$ De plus, il est opéré en utilisant $N$ fonctions de hachage $\{f_1, f_2, \ldots, f_{N-1}, f_N\}$. Chacune de ces fonctions est définie comme une injection $X \to \{0, \ldots, M-1\}$. Pour insérer un objet $x \in X$ dans un filtre de Bloom, une liste de $N$ bits est obtenue. Cette liste est $\langle f_1(x), f_2(x), \ldots, f_{N-1}(x), \ldots, f_N(x)\rangle$. Chacun de ces bits est mis à 1. Pour vérifier la présence d'un objet $x$ dans un filtre de Bloom, on obtient la liste des bits $\langle f_1(x), f_2(x), \ldots, f_{N-1}(x), \ldots, f_N(x)\rangle$. L'objet n'est définitivement pas présent dans le filtre de Bloom si au moins un des bits de l'objet $x$ est à 0. Sinon, l'objet est considéré comme étant dans le filtre de Bloom, avec un certain taux de faux positifs. Dans Ray, chaque rang MPI utilise un filtre de Bloom pour filtrer les k-mers contenant des erreurs de séquençage.

Dans Ray, le filtre de Bloom est adaptatif et distribué. Chaque rang MPI possède son propre filtre de Bloom. Le filtre de Bloom de chaque rang MPI est adaptatif car son nombre de bits disponibles est une fonction linéaire du nombre de séquences d'ADN qui sont stockées sur le rang MPI. En effet, le nombre de $k$-mers sur chaque rang MPI est une fonction linéaire du nombre de séquences d'ADN. [60]. Lorsqu'une erreur de substitution survient dans le milieu d'une séquence d'ADN, il s'ensuit que les $k$ $k$-mers qui contiennent la position avec l'erreur seront peu abondants. Un filtre de Bloom pourra alors les éliminer à la source si ces $k$-mers sont observés une seule fois.

Pour réduire encore plus l'utilisation de la mémoire, le chaînage de filtres de Bloom (similaire à la cascade décrite dans [258]) pourrait être envisagé. Par exemple, le chaînage de deux filtres de Bloom permet d'éliminer les objets qui surviennent une ou deux fois [258].

De plus, le filtre de Bloom peut aussi être utilisé pour obtenir une représentation probabiliste d'un graphe de de Bruijn [213]. Dans l'article [213], les auteurs ont utilisé un filtre de Bloom pour stocker la présence des $k$-mers dans un ensemble de lectures d'ADN. Cette représentation est probabiliste à cause que le filtre de Bloom a un taux de faux positifs supérieur à 0.

### 4.8.4 Table de hachage clairsemée distribuée

Ray utilise une table de hachage implémentée en C++ 1998 (en tant que gabarit générique) dans RayPlatform. Le code est distribué avec la version 3 de la licence publique générale limitée GNU (LGPLv3 – *Lesser GNU General Public License, version 3*). La table de hachage est distribuée sur $n$ rangs MPI. La fonction $h_3 : X \to \{0, 1, 2, \ldots, n-2, n-1\}$ détermine sur quel rang MPI doit aller l'objet $x \in X$. Ici, l'ensemble d'objets $X$ est l'ensemble de tous les mots de longueur $k$ possibles à partir de l'alphabet $\Sigma = \{A, T, C, G\}$.

#### Table de hachage

Une table de hachage avec adressage ouvert contient $N$ seaux (en anglais : *bucket*) pour y mettre un maximum de $N$ objets [139]. Une table peut donc adresser sans collision jusqu'à $N$ objets différents. Pour obtenir la position $i \in \{0, 1, 2, \ldots, N-2, N-1\}$ d'un objet $x \in X$ dans la table de hachage, il faut utiliser une fonction qui dit dans quel seau doit aller un objet. Cette fonction est une injection telle que $h_1 : X \to \{0, 1, 2, \ldots, N-2, N-1\}$. Les objets $x, y \in X$ sont en collision si $h_1(x) = h_1(y)$, c'est-à-dire s'ils vont dans le même seau. Ici, la table de hachage utilise l'adressage ouvert pour gérer les collisions. Plus précisément, le hachage double est utilisée. Pour ce faire, une deuxième fonction de hachage $h_2 : X \to \{1, 3, 5, \ldots, N-5, N-3, N-1\}$ est utilisée. $h_2$ a la propriété qu'elle retourne une image qui est co-première avec $N$ (le nombre total de seaux). En pratique, $N$ est une puissance de 2, et donc un nombre pair, et $h_2$ retourne un nombre impair inclusivement entre 1 et $N-1$. Cette astuce est la façon la plus facile d'avoir des nombres co-premiers. L'autre façon est d'utiliser uniquement des nombres premiers pour $N$, ce qui est plus difficile.

#### Hachage double

Le principe du hachage double est de calculer le seau primaire en utilisant seulement $h_1$. Le numéro de la première opération de sondage (en anglais : *probe operation*) est $q = 0$. Lorsque l'opération de sondage retourne une collision, une autre opération de sondage

est faite, en utilisant $q \leftarrow q + 1$. Dans le pire des cas, tous les seaux sont sondés, ce qui génère $N$ opérations de sondage. Le hachage double regarde donc en premier dans le premier seau pour voir s'il est libre. S'il est occupé par un objet différent, alors un autre sceau est obtenu en utilisant le hachage double en utilisant cette fois-ci $h_1$ et $h_2$.

À partir des fonctions $h_1$ et $h_2$, et avec un numéro d'opération de sondage $q \in \{0, 1, 2, 3, \ldots, N-3, N-2, N-1\}$, la fonction de hachage $h^* : X \to \{0, 1, \ldots, N-2, N-1\}$ est définie récursivement par

$$h^*(x,q) = \begin{cases} h_1(x) & \text{si } q = 0 \\ h^*(x, q-1) - h_2(x) & \text{sinon si } h^*(x, q-1) - h_2(x) \geq 0 \\ h^*(x, q-1) - h_2(x) + N & \text{sinon si } h^*(x, q-1) - h_2(x) < 0 \end{cases} \tag{4.5}$$

L'équation 4.5 peut être écrite plus simplement (mais donne des valeurs différentes) avec $h^*(x,q) = (h_1(x) + q\ h_2(x))\ mod\ N$ où $h^*$ est la fonction de hachage double, $x$ est l'objet, $q$ est le numéro de l'opération (entre 0 et $N-1$ inclusivement), $h_1$ et $h_2$ sont des fonctions de hachage et $N$ est le nombre de seaux (la taille de la table).

**Redimensionnement incrémentiel amorti**

Étant donné que les objets sont mis dans des seaux en utilisant le hachage double, il est impossible d'ajouter des seaux. Une façon de procéder est de créer une table auxiliaire deux fois plus grande, avec $N \leftarrow N + N$, et d'y transférer tous les objets avec des opérations amorties. Ce processus s'appelle le redimensionnement incrémentiel amorti. Le modèle de stockage du tableau d'objets est clairsemé, c'est-à-dire qu'une position libre utilise quelques bits au lieu de la taille d'un objet. La structure de tableau a été décrite par les auteurs du *Google Sparse Hash*. Les seaux sont répartis dans des groupes contenant chacun $M$ seaux. Pour des raisons algorithmiques, $N$ est une puissance de 2. Le tableau contient $N$ seaux, qui sont répartis dans $N/M$ seaux. Le seau $i \in \{0, 1, \ldots, N-2, N-1\}$ est géré par le groupe $g = i/M$ alors que le numéro du seau à l'intérieur du groupe $j \in \{0, 1, 2, \ldots, M-3, M-2, M-1\}$ est calculé avec $j = i\%M$ où % est le reste de la division entière.

La conception hiérarchique de la table permet d'obtenir une performance exceptionnelle en mémoire. Finalement, la table de hachage dans Ray utilise un allocateur de mémoire spécialisé qui utilise des pointeurs intelligents [79] ainsi qu'un compacteur de mémoire avec des opérations amorties.

### 4.8.5 Stockage compact des sommets

Chacun des deux sommets dans toutes les paires de sommets de longueur impaire peuvent être stocké ensemble. Premièrement, la fonction d'encodage de nucléotide est définie par

$$enc(\sigma) = \begin{cases} \langle 0,0\rangle & \sigma = A \\ \langle 0,1\rangle & \sigma = C \\ \langle 1,0\rangle & \sigma = G \\ \langle 1,1\rangle & \sigma = T \end{cases}. \tag{4.6}$$

Pour obtenir le complément inverse d'une séquence d'ADN, il faut inverser l'ordre des symboles et ensuite obtenir le complément de chaque symbole. Le complément d'un symbole est défini comme

$$\text{comp}(\sigma) = \begin{cases} A & \sigma = T \\ T & \sigma = A \\ C & \sigma = G \\ G & \sigma = C \end{cases} \tag{4.7}$$

L'avantage de l'encodage est que $enc(comp(\sigma)) = \tilde{} enc(\sigma)$ où $\tilde{}$ est l'opérateur de négation qui change les 0 pour des 1 et les 1 pour des 0. De manière générale, le complément inverse de la séquence

$s = \langle s_1, s_2, s_3, \ldots, s_{|s|-3}, s_{|s|-2}, s_{|s|-1}, s_{|s|}\rangle$

est $\overline{s} = \langle comp(s_{|s|}), comp(s_{|s|-1}), comp(s_{|s|-2}), comp(s_{|s|-3}), \ldots,$
$comp(s_3), comp(s_2), comp(s_1)\rangle$.

Pour stocker une séquence et son complément inverse au même endroit dans la mémoire, il faut s'assurer qu'aucune séquence est son propre complément inverse. Ce comportement peut être atteint en utilisant des séquences de longueur impaire. Pour une longueur de mots $k = 2t + 1$ qui est impaire, et l'alphabet $\Sigma\{A, T, C, G\}$, aucun mot $x \in \Sigma^k$ ne peut être son propre complément inverse.

Supposons que $s$ et $\overline{s}$ sont des compléments inverses, que ce sont des mots de longueur impaire, et qu'ils sont identiques. Puisque les mots sont de longueur impaire $k$, la position du milieu de n'importe quel mot est $t = \lfloor k/2 \rfloor + 1$. Par définition, $s_t = \overline{s}_t$ ($s$

et $\overline{s}$ sont identiques) et aussi $\overline{s}_t = comp(s_t)$, ce qui est une contradiction. Il est donc impossible d'avoir un mot qui est son propre complément inverse si $k$ est impaire.

Dans Ray, tous les mots sont entreposés en paires. Pour une paire de mots compléments inverses $s$ et $\overline{s}$, le représentant de la paire est obtenue par

$$\alpha(s, \overline{s}) = \begin{cases} s & s < \overline{s} \\ \overline{s} & s > \overline{s} \end{cases} . \tag{4.8}$$

Il est donc possible de stocker tous les mots en entreposant seulement la moitié d'entre eux. De plus, seulement 2 bits par nucléotide sont nécessaires.

### 4.8.6 Stockage compact des arêtes

Les parents et les enfants d'un sommet dans le graphe de de Bruijn peuvent être encodés dans 8 bits puisqu'il y a un maximum de 4 parents (A, T, C, G) et un maximum de 4 enfants (A, T, C, G) et que l'état de ces arcs est binaire. Pour un sommet quelconque $s \in \Sigma^k$, il y a au plus 4 parents et au plus 4 enfants dans les données de séquençage. Pour représenter les arêtes du sommet $s$, il faut une carte de 8 bits, notée par $\langle b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0 \rangle$. Les bits de 0 à 3 sont les parents et les bits de 4 à 7 sont les enfants. Si le sommet $ATGAC$ a 1 parent $\{GATGA\}$ et deux enfants $\{TGACT, TGACA\}$, alors sa carte contient 3 bits à 1 et les autres à 0. La Figure 4.4 à la page 60 illustre cet exemple.

Ici aussi, il est donc possible de stocker les arêtes de tous les sommets en utilisant seulement la moitié de l'espace nécessaire.

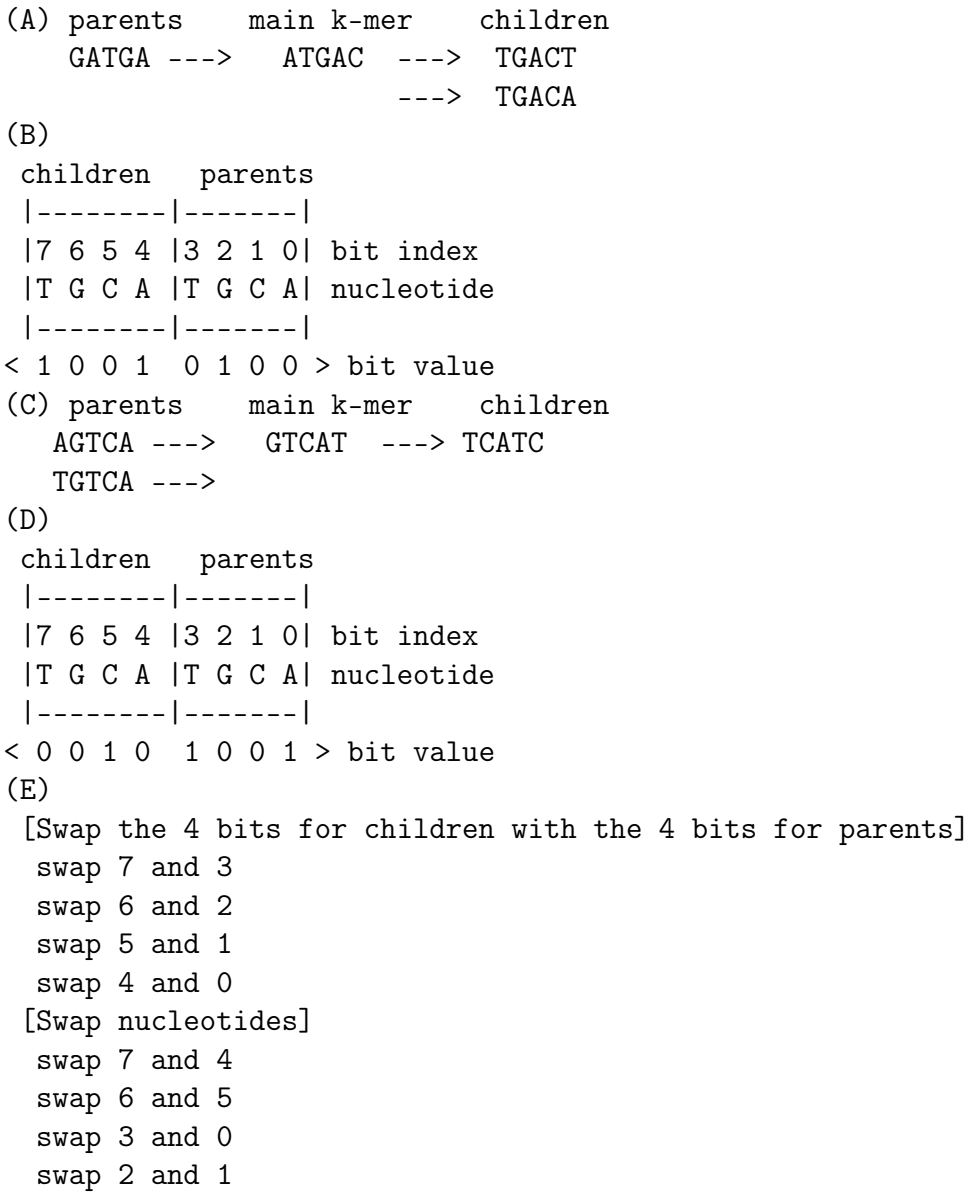


FIGURE 4.4: Exemple de carte de 8 bits pour les arêtes.

(A) Un sommet et ses voisins. (B) La carte pour le sommet en (A). (C) Le complément inverse du sommet en (A). (D) La carte pour le sommet en (C). (E) L'algorithme pour passer de (B) à (D).

## 4.9 Particularités de Ray pour l'assemblage

### 4.9.1 Heuristiques d'exploration

Le chapitre 6 contient un article publié dans le *Journal of Computational Biology*. Dans cet article, les heuristiques sont décrites en détail. Les extensions Ray par-dessus RayPlatform pour faire l'assemblage *de novo* sont toutes implémentées en C++ 1998 en utilisant l'API (*Application Programming Interface*) de RayPlatform. Ray construit un graphe distribué, c'est-à-dire que les sommets sont entreposés sur plusieurs rangs MPI. L'entreposage est non redondant. La séquence du génome est un chemin dans ce graphe, qui est un sous-graphe du graphe de de Bruijn. Avec plusieurs types de données, Ray calcule des chemins dans ce graphe. Pour s'y orienter, des heuristiques sont utilisés. Il y a des heuristiques pour les séquences en paires, pour les séquences seules, et pour les séquences éloignées de type *mate pairs*. Les heuristiques sont décrites dans [30].

### 4.9.2 Assemblage *de novo* de jeux énormes de données

Dans notre article publié dans *Genome Biology* au mois de décembre 2012 [32] (voir le chapitre 7), un méta-génome contenant 3 milliards de séquences de 100 nucléotides a été assemblé *de novo* avec le logiciel Ray et les formes de vie ont été quantifiées de manière très précise. Ce méta-génome contenait 1000 génomes bactériens – avec des souches très similaires. Ce calcul a été fait en environ 15 heures en utilisant 1024 cœurs de processeur. Pour ce faire, 1024 processus ont été exécutés de manière distribuée et ceux-ci ont communiqué en utilisant le passage de messages (voir le chapitre 3).

### 4.9.3 Modifications aux algorithmes de Ray pour la méta-génomique

Dans l'article décrivant les heuristiques de Ray [30] (chapitre 6), il était assumé que la distribution du signal pour les mots dans le sous-graphe du graphe de de Bruijn était globale et pouvait résumer tous les événements (par exemple, on pouvait supposer que l'observation d'une séquence lorsqu'on regarde un à un toutes les lectures d'ADN suivait une loi de Poisson ). Pour les transcriptomes et les méta-génomes, la distribution du signal possède de fortes caractéristiques de localité – c'est-à-dire que la distribution du graphe au complet est en fait une distribution sur l'union de graphes qui sont probablement indépendants dans la nature. Chacun de ces graphes représente un des organismes dans le microbiome. Il n'est pas clair à quoi ressemble cette distribu-

tion jointe. Ces abondances non-uniformes dans les méta-génomes empêche l'utilisation directe des assembleurs *de novo* classiques qui sont conçus pour les génomes individuels.

Afin d'être en mesure d'assembler des méta-génomes avec Ray, il a été nécessaire de généraliser plusieurs parties importantes dans les modules d'extension de Ray (des modules d'extension qui sont 100% compatible avec l'engin de calcul distribué Ray-Platform décrit dans la section 3.6) afin d'exploiter cette localité. Les changements apportés sont présentés dans le chapitre 7. Parmi les changements, le calcul des marqueurs de lectures se fait maintenant en utilisant seulement les valeurs de profondeur de séquençage des sommets d'une lecture donnée et ne dépend pas de la distribution globale de la profondeur de séquençage. Un autre changement notable est qu'une distribution des profondeurs de séquençage est calculée individuellement pour chaque amorce d'assemblage [32]. Dans l'assembleur Ray initial, il n'y avait qu'une seule distribution des profondeurs de séquençage pour l'ensemble du sous-graphe.

### 4.9.4 Calcul d'abondances taxonomiques

Les taxons peuvent être quantifiés en utilisant un dénombrement exhaustif de mots génétiques qui y sont associés. Ces mots génétiques peuvent être des sous-séquences du génome. Environ au même moment que la date de publication de notre article dans *Genome Biology* [32], l'équipe de Ross Overbeek du *Fellowship for the Interpretation of Genomes* a publié une preuve de concept que le dénombrement de mots spécifiques pouvait aussi se faire à l'intérieur de l'espace des protéines [80]. Cette dernière publication du groupe de Overbeek a mis l'accent sur l'aspect en temps réel, comme dans la publication [243].

Afin de quantifier des taxons en utilisant un dénombrement de mots, il est nécessaire d'utiliser une classification reconnue des objets biologiques. Pour le profilage taxonomique, par exemple, une base de données de génomes connus est habituellement utilisée pour identifier les séquences d'un échantillon [187, 9]. Le logiciel *lmat* [9] est implémenté en C++ utilise un arbre taxonomique (la taxonomie du *National Center for Biotechnology Information*), une base de données de génomes de référence et finalement des associations entre les génomes et les taxons [9]. Le flux d'analyse *Ray Communities* utilise exactement les mêmes entrées pour calculer des abondances taxonomiques [32].

Des gènes particuliers (par exemple, le gène qui encode la partie 16S du ribosome) appelés marqueurs sont utilisés par des outils qui calculent également des abondances taxonomiques. Les outils MetaPhlAn [271], MetaPhyler [165], Phymm [35] et PhymmBL

[35] font parti de cette liste de logiciels utilisant des marqueurs plutôt que des génomes complets. Ces outils qui utilisent des marqueurs sont moins précis [9], mais sont plus rapides.

## 4.10 Visualisation interactive d'un graphe de de Bruijn

Après avoir rencontré plusieurs situations nécessitant une compréhension structurelle de graphe de de Bruijn pour des projets particuliers – par exemple, pour un projet d'amélioration de la qualité des ancres d'assemblage, j'ai eu l'idée de développer un outil offrant une expérience visuelle interactive de graphes de de Bruijn. Le logiciel que j'ai créé pour cette tâche s'appelle Ray Cloud Browser et le code source est disponible à l'adresse
https://github.com/sebhtml/Ray-Cloud-Browser.

J'ai présenté mes travaux de visualisation avec Ray Cloud Browser à l'Université de Liverpool le 27 novembre 2013, au Royaume-Uni. La vidéo de ma présentation est sur le site YouTube :
https://www.youtube.com/watch?v=tSul_qDwvN4#t=5m46s.

### 4.10.1 Motivation et applications

L'assemblage de données génétiques commence dans certains cas avec un graphe de de Bruijn. Le graphe de de Bruijn est plus gourmand en mémoire que le graphe à chaîne, mais ce désavantage est caché par l'information beaucoup plus granulaire offerte par un tel graphe (par exemple, sa capacité à associer une profondeur de séquençage à chaque sommet). Il n'est pas facile de visualiser le graphe de de Bruijn généré pour un échantillon, car il a énormément de sommets. Une approche utilisée, mais qui génère une expérience statique est l'outil dot du projet Graphviz [94]. Il peut y avoir plusieurs raisons pour vouloir visualiser un tel graphe. Une de celles-ci est de mieux comprendre les algorithmes et méthodes d'assemblage *de novo*. J'ai donc développé Ray Cloud Browser, un visualiseur interactif de graphes de de Bruijn pour la génomique. Voici quelques exemples de cas d'utilisation. Bien sûr, Ray Cloud Browser étant un outil interactif et dynamique, les images ci-dessous ne montrent pas cet aspect du logiciel puisqu'elles sont statiques. La figure 4.5 montre une rupture présente entre deux chemins dans une des étapes d'assemblage fait par le logiciel Ray.

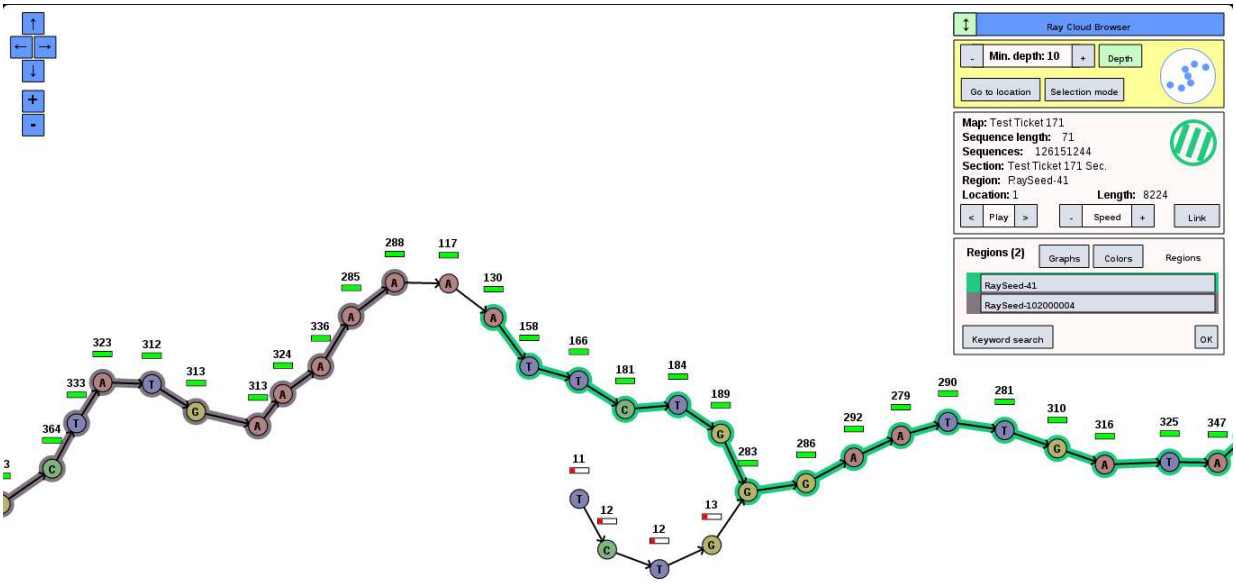


FIGURE 4.5: Rupture de séquence entre deux chemins.

Le chemin mauve (sur le côté gauche) et le chemin vert (sur le côté droit) ne sont pas connectés. Cependant, la géométrie de la structure indique que ces deux chemins devraient être liés. Les nombres au dessus des sommets sont les profondeurs de séquençage. Les barres au dessus des sommets (verte ou rouge dans la figure) représente visuellement la profondeur de séquençage.

Ray Cloud Browser peut aussi être utilisé pour faire le contrôle de la qualité de données de séquençage. Nous avons confirmé visuellement les biais dans les erreurs de séquençage tel que reporté dans la littérature [63, 103, 119] (voir aussi la section 1.2.6) en utilisant Ray Cloud Browser. La figure 4.6 montre un système de régions répétées dans lequel sont présentes des impasses.

Le dernier cas d'utilisation que je désire montrer est une région répétée utilisée par trois régions génomiques. Cette mise en scène de plusieurs régions est affichée dans la figure 4.7.

### Projets existants

Il existe quelques projets pour la visualisation de données biologiques mais aucun ne vise la visualisation de graphe de de Bruijn utilisés pour l'assemblage *de novo*. Parmi les logiciels de visualisation disponibles, Phylo est un jeu pour faire des alignements visuellement [137]. Certains jeux comme Phylo ont un but scientifique [99]. Phylo permet d'améliorer les alignements qui sont trop difficiles pour les algorithmes usuels. ABySS-Explorer est un logiciel de visualisation d'assemblages [207, 206]. Dans ABySS-Explorer, les séquences assemblées sont représentées comme des ressorts dont le nombre

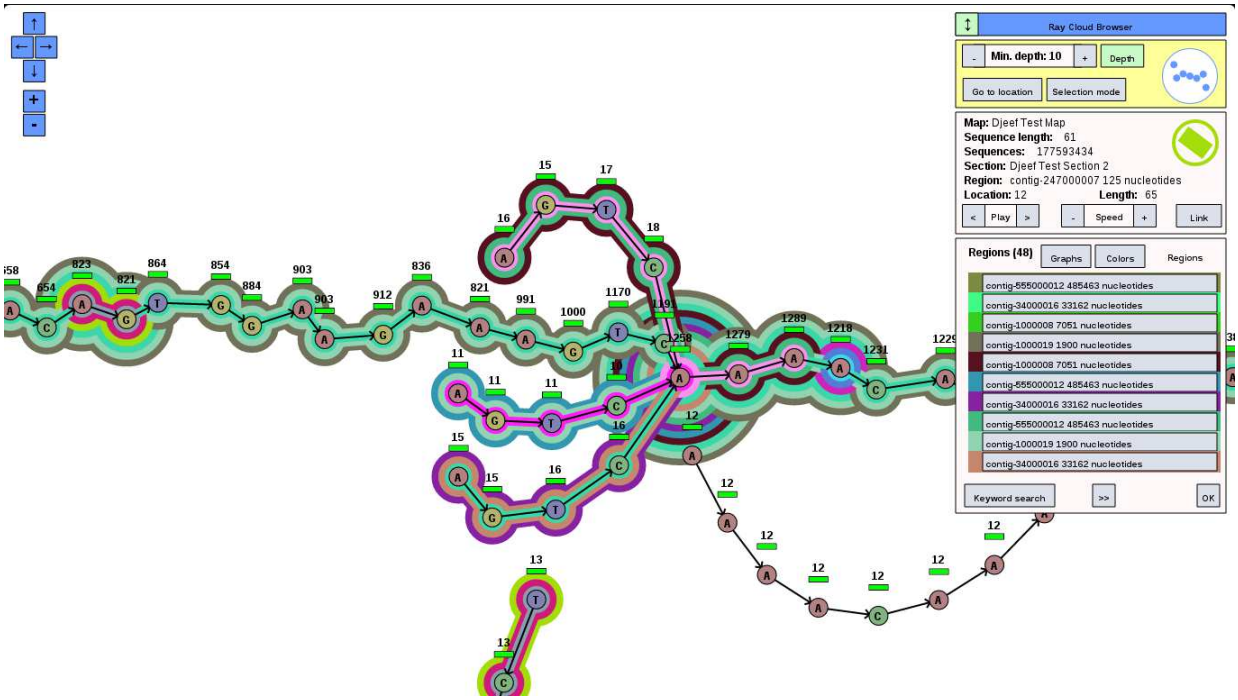


FIGURE 4.6: Visualisation d'impasses dans le logiciel Ray Cloud Browser.

Les impasses sont visibles et ont une profondeur de séquençage plus basse que le reste de la structure génétique. Sur cette figure, les impasses sont les chemins qui ne sont pas liés aux deux extrémités.

d'oscillations est proportionnel au nombre de nucléotides de la séquence assemblée. Une limitation de ABySS-Explorer est que le logiciel n'est pas libre et que le code source n'est pas disponible. De plus, ce n'est pas un logiciel qui peut s'exécuter directement à partir d'une page web. MGAviewer est un logiciel de visualisation d'alignements pour la méta-génomique [310]. Ce logiciel permet, dans son essence, de visualiser des alignements entre les lectures et des génomes de référence. Scribl est une librairie pour la visualisation de données génomiques sur le web [188]. Cependant, Scribl n'est pas interactif, et n'est pas non plus un outil de visualisation. En effet, Scribl est une librairie qui fournit des façons de dessiner des objets biologiques comme les gènes et les protéines. ChromoZoom est un navigateur de génome pour le web [211]. Comme son nom l'indique, ChromoZoom permet de faire des rétrécissements ou encore des agrandissements de chromosomes. Contrairement aux outils énumérés ci-haut, le rendu graphique dans Ray Cloud Browser est vectoriel alors qu'il est matriciel dans les autres logiciels. Un rendu vectoriel permet d'utiliser des facteurs d'agrandissement et de rétrécissement tout en gardant la même qualité de dessin.

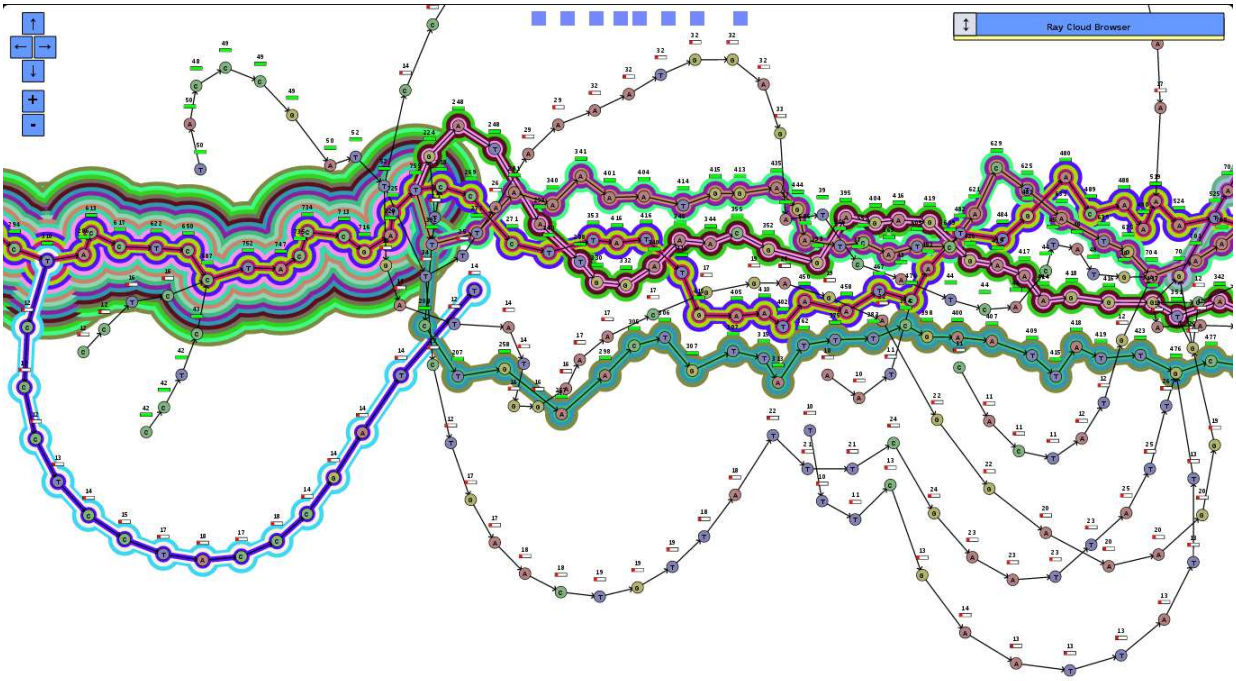


FIGURE 4.7: Région répétée utilisée par quatre séquences génomiques.

Dans cette illustration, une région répétée est réutilisée par quatre séquences assemblées rapportées par Ray.

### 4.10.2 Implémentation

Le premier critère pour les choix technologiques a été la portabilité. L'explorateur infonuagique Ray Cloud Browser utilise le passage de messages sur le protocole HTTP – *Hyper Text Transfer Protocol* – afin d'exploiter une architecture client-serveur. La version du protocole utilisée est HTTP/1.1. Le client et le service web sont construits autour du web avec le protocole HTTP.

#### Client

L'architecture est distribuée et caractérisée par un ou plusieurs clients et un serveur. Le client qui est distribué à titre d'exemple est codé en ECMA Javascript et utilise HTML5 (*Hyper Text Markup Language* 5) pour le rendu visuel. Plus particulièrement, la balise *canvas* est utilisée. Cette balise permet de faire des opérations de dessin comme dessiner une ligne et dessiner un cercle.

#### Ressources du service web

Le service web est implémenté en C++ 1998 avec une interface HTTP exposée publiquement. La méthode HTTP GET est utilisée pour obtenir des informations à partir

du service web. Un connecteur unique sert d'interface avec laquelle les clients peuvent interagir. Le tout est déployable dans les nuages, par exemple le service de nuage élastique de calcul offert par la compagnie Amazon Web Services, Inc. (Amazon EC2, *Elastic Compute Cloud*). Nous avons des déploiements en production dans les nuages chez Amazon Web Services, Inc. sur des instances de type t1.micro dans le marché des enchères (*spot instance market*). Les instances t1.micro ont 613 mégaoctets de mémoire et très peu de cycles de processeur. Les instances m1.small ont 1.7 Gio de mémoire, 1 cœur virtuel, 160 Gio de stockage d'instance sur un périphérique de blocs éphémère et un volume de stockage de bloc élastique (EBS pour *Elastic Block Storage*) pour le système d'exploitation qui utilise un noyau monolithique Linux. Le modèle des services web n'est pas nouveau en biologie. Par exemple, le projet *SEED* utilise des services web [70, 14, 187, 192]. L'avantage des services web est que l'interface pour opérer les systèmes d'analyse est bien définie. Par contre, un service web nécessite une infrastructure de provisionnement de machines virtuelles, ce qui peut coûter cher.

**Formats de fichier natifs**

Le service web utilise trois types de format de fichier qui permettent d'entreposer les cartes génétiques, les sections, et les annotations. Ces formats sont compacts et efficaces. Ces formats permettent de représenter des cartes, des sections, des régions et des emplacements dans un génome. Le graphe est entreposé dans un fichier binaire et l'extraction d'information est faite avec une recherche binaire. L'algorithme utilisé est simplement une recherche binaire. De plus, pour trouver les séquences avec des annotations, l'indice de chacune des séquences est utilisée comme clé. D'autres algorithmes existent pour trouver les intersections entre des jeux de données. Un de ces algorithmes est le *Binary Interval Search* décrit par Layer *et al.* en 2012 [151].

# Chapitre 5

# Article 1 : HIV-1 coreceptor usage prediction without multiple alignments : an application of string kernels

## 5.1 Journal

Ce manuscrit a été publié dans le journal *Retrovirology* le 2008-12-04 pendant ma maîtrise et est inclus dans ma thèse, car je n'ai pas écrit de mémoire de maîtrise. 

## 5.2 Résumé

Contexte :

Le virus de l'immunodéficience humaine de type 1 (VIH-1) infecte les cellules par l'intermédiaire d'interactions ligand-récepteur. Ce lentivirus utilise le récepteur CD4 en combinaison avec un corécepteur des chimiokines, soit CXCR4 ou CCR5, pour entrer dans une cellule cible. VIH-1 est caractérisé par une variabilité de séquence élevée. Néanmoins, dans cette grande variabilité, certaines fonctionnalités doivent être conservées pour définir les fonctions et les phénotypes. La détermination du corécepteur exploité par VIH-1, à partir de sa séquence d'enveloppe protéique, tombe dans le domaine de l'apprentissage automatique. Le problème est connu sous le nom de classification. La

machine à vecteurs de support (SVM), avec les noyaux à chaînes (*string kernels* en anglais), s'est avéré très efficace pour faire face à une large classe de problèmes de classification allant de la catégorisation de textes à la détection d'homologie des protéines. Dans cet article, nous examinons comment le SVM peut prévoir l'utilisation de corécepteur quand il est équipé d'un noyau à chaîne approprié. En ce sens, nous y présentons également un nouveau noyau à chaînes, le noyau à segments éloignés (en anglais, *Distant segments kernel*) qui s'avèrera particulièrement bien approprié pour cette tâche.

Résultats :

Trois noyaux ont été comparés. Des précisions de 96,35% (CCR5) 94.80% (CXCR4) et 95,15% (CCR5 et CXCR4) ont été réalisées avec le SVM équipé du noyau à segments éloignés sur un ensemble de test de 1425 exemples avec un classificateur construit sur un ensemble d'apprentissage de 1425 exemples. Nos bases de données sont construites avec la bases de données des séquences de VIH de Los Alamos National Laboratory. Un serveur Web est disponible à l'adresse http ://genome.ulaval.ca/hiv-dskernel.

Conclusions :

Nous avons examiné les noyaux à chaînes qui ont été utilisés avec succès pour la détection d'homologie de protéines et avons proposé un nouveau noyau que nous appelons le noyau à segments éloignés. Nous avons également montré comment extraire les caractéristiques les plus pertinentes pour l'utilisation du corécepteur du VIH-1. Le SVM avec le noyau à segments éloignés est actuellement la meilleure méthode décrite.

## 5.3 Contributions

J'ai écrit le brouillon du manuscrit. J'ai fait la preuve avec François Laviolette pour le noyau à segments distants. J'ai écrit tout le code et fait toutes les expériences. La section 5.12 contient les contributions des auteurs.

# HIV-1 coreceptor usage prediction without multiple alignments: an application of string kernels

Sébastien Boisvert[1] Mario Marchand[2]
François Laviolette[2] and Jacques Corbeil*[1]

2008-12-04

(1)Centre de recherche du centre hospitalier de l'Université Laval, Québec (QC), Canada
(2)Département d'informatique et de génie logiciel, Université Laval, Québec (QC), Canada

## 5.4 Abstract

**Background:** Human immunodeficiency virus type 1 (HIV-1) infects cells by means of ligand-receptor interactions. This lentivirus uses the CD4 receptor in conjunction with a chemokine coreceptor, either CXCR4 or CCR5, to enter a target cell. HIV-1 is characterized by high sequence variability. Nonetheless, within this extensive variability, certain features must be conserved to define functions and phenotypes. The determination of coreceptor usage of HIV-1, from its protein envelope sequence, falls into a well-studied machine learning problem known as *classification*. The support vector machine (SVM), with string kernels, has proven to be very efficient for dealing with a wide class of classification problems ranging from text categorization to protein homology detection. In this paper, we investigate how the SVM can predict HIV-1 coreceptor usage when it is equipped with an appropriate string kernel.

**Results:** Three string kernels were compared. Accuracies of 96.35% (CCR5) 94.80% (CXCR4) and 95.15% (CCR5 and CXCR4) were achieved with the SVM equipped with the *distant segments kernel* on a test set of 1425 examples with a classifier built on a training set of 1425 examples. Our datasets are built with Los Alamos National Laboratory HIV Databases sequences. A web server is available at http://genome.ulaval.ca/hiv-dskernel.

**Conclusions:** We examined string kernels that have been used successfully for protein homology detection and propose a new one that we call the *distant segments kernel*. We also show how to extract the most relevant features for HIV-1 coreceptor usage. The SVM with the *distant segments kernel* is currently the best method described.

## 5.5 Background

The HIV-1 genome contains 9 genes. One of the genes, the *env* gene, codes for 2 envelope proteins named gp41 and gp120. The gp120 envelope protein must bind to a CD4 receptor and a coreceptor prior to cell infection by HIV-1. Two coreceptors can be used by HIV-1: the CCR5 (chemokine receptor 5) and the CXCR4 (chemokine receptor 4). Some viruses are only capable of using the CCR5 coreceptor. Other viruses can only use the CXCR4 coreceptor. Finally, some HIV-1 viruses are capable of using both of these coreceptors. The pathology of a strain of HIV-1 is partly a function of the coreceptor usage [227]. The faster CD4+ cell depletion caused by CXCR4-using viruses

[242] makes the accurate prediction of coreceptor usage medically warranted. Specific regions of the HIV-1 external envelope protein, named hypervariable regions, contribute to the turnover of variants from a phenotype to another [309]. HIV-1 tropisms (R5, X4, R5X4) are often (but not always) defined in the following way. R5 viruses are those that can use only the CCR5 coreceptor and X4 viruses are those that can use only the CXCR4 coreceptor. R5X4 viruses, called dual-tropic viruses, can use both coreceptors. Tropism switch occurs during progression towards AIDS. Recently, it has been shown that R5 and X4 viruses modulate differentially host gene expression [277].

### 5.5.1 Computer-aided prediction

The simplest method used for HIV-1 coreceptor usage prediction is known as the *charge rule* [5, 6]. It relies only on the charge of residues at positions 11 and 25 within the V3 loop aligned against a consensus. The V3 loop is the third highly variable loop in the retroviral envelope protein gp120. Nonetheless, other positions are also important since the removal of these positions gave predictors with comparable (but weaker) performance to those that were trained with these positions present [227]. Other studies [241, 8, 131, 261, 304, 12] also outlined the importance of other positions and proposed machine learning algorithms, such as the random forest [304] and the support vector machine (SVM) with structural descriptors [261], to built better predictors (than the charge rule). Available predictors (through webservers) of HIV-1 coreceptor usage are enumerated in [153].

An accuracy of 91.56% for the task of predicting the CXCR4 usage was obtained by [261]. Their method, based on structural descriptors of the V3 loop, employed a single dataset containing 432 sequences without indels and required the multiple alignment of all V3 sequences. However, such a prior alignment before learning might remove information present in the sequences which is relevant to the coreceptor usage task. Furthermore, a prior multiple alignment done on all the data invalidates the cross-validation method since the testing set in each fold has been used for the construction of the tested classifier. Another drawback of having an alignment-based method is that sequences having too many indels (when compared to a consensus sequence) are discarded to prevent the multiple alignment from yielding an unacceptable amount of gaps. In this paper, we present a method for predicting the coreceptor usage of HIV-1 which does not perform any multiple alignment prior to learning.

The SVM [14] has proven to be very effective at generating classifiers having good gen-

eralization (i.e., having high predicting accuracy). In particular, [227] have obtained a significantly improved predictor (in comparison with the charge rule) with an SVM equipped with a linear kernel. However, the linear kernel is not suited for sequence classification since it does not provide a natural measure of dissimilarity between sequences. Moreover, a SVM with a linear kernel can only use sequences that are exactly of the same length. Consequently, [227] aligned all HIV-1 V3 loop sequences with respect to a consensus. No such alignment was performed in our experiments. In contrast, string kernels [15] do not suffer from these deficiencies and have been explicitly designed to deal with strings and sequences of varying lengths. Furthermore, they have been successfully used for protein homology detection [255] — a classification problem which is closely related to the one treated in this paper.

Consequently, we have investigated the performance of the SVM, equipped with the appropriate string kernel, at predicting the coreceptor used by HIV-1 as a function of its protein envelope sequence (the V3 loop). We have compared two string kernels used for protein homology detection, namely the blended spectrum kernel [154, 15] and the local alignment kernel [255], to a newly proposed string kernel, that we called the *distant segments* (DS) kernel.

### 5.5.2 Applications

Bioinformatic methods for predicting HIV phenotypes have been tested in different situations and the concordance is high [18, 19, 20, 21].

As described in [18], current bioinformatics programs are underestimating the use of CXCR4 by dual-tropic viruses in the brain. In [19], a concordance rate of 91% was obtained between genotypic and phenotypic assays in a clinical setting of 103 patients. In [20], the authors showed that the SVM with a linear kernel achieves a concordance of 86.5% with the Trofile assay and a concordance of 79.7% with the TRT assay. Recombinant assays (Trofile and TRT) are described in [20].

Further improvements in available HIV classifiers could presumably allow the replacement of in vitro phenotypic assays by a combination of sequencing and machine learning to determine the coreceptor usage. DNA sequencing is cheap, machine learning technologies are very accurate whereas phenotypic assays are labor-intensive and take weeks to produce readouts [153]. Thus, the next generation of bioinformatics programs for the prediction of coreceptor usage promises major improvements in clinical settings.

## 5.6 Methods

We used the SVM to predict the coreceptor usage of HIV-1 as a function of its protein envelope sequence. The SVM is a discriminative learning algorithm used for binary classification problems. For these problems, we are given a *training set* of *examples*, where each example is labelled as being either *positive* or *negative*. In our case, each example is a string $s$ of amino acids. When the binary classification task consists of predicting the usage of CCR5, the label of string $s$ is $+1$ if $s$ is the V3 loop of the protein envelope sequence of a HIV-1 virion that uses the CCR5 coreceptor, and $-1$ otherwise. The same method applies for the prediction of the CXCR4 coreceptor usage. When the binary classification task consists of predicting the capability of utilizing CCR5 and CXCR4 coreceptors, the label of string $s$ is $+1$ if $s$ is the V3 loop of the protein envelope sequence of a HIV-1 virion that uses both the CCR5 and CXCR4 coreceptors, and $-1$ if it is a virion that does not use CCR5 or does not use CXCR4.

Given a training set of binary labelled examples, each generated according to a fixed (but unknown) distribution $D$, the task of the learning algorithm is to produce a classifier $f$ which will be as accurate as possible at predicting the correct class $y$ of a test string $s$ generated according to $D$ (*i.e.*, the same distribution that generated the training set). More precisely, if $f(s)$ denotes the output of classifier $f$ on input string $s$, then the task of the learner is to find $f$ that minimizes the probability of error $\Pr_{(s,y)\sim D}(f(s) \neq y)$. A classifier $f$ achieving a low probability of error is said to *generalize* well (on examples that are not in the training set).

To achieve its task, the learning algorithm (or learner) does not have access to the unknown distribution $D$, but only to a limited set of training examples, each generated according to $D$. It is still unknown exactly what is best for the learner to optimize on the training set, but the learning strategy used by the SVM currently provides the best empirical results for many practical binary classification tasks. Given a training set of labelled examples, the learning strategy used by the SVM consists at finding a soft-margin hyperplane [14, 290], in a feature space of high dimensionality, that achieves the appropriate trade-off between the number of training errors and the magnitude of the separating margin realized on the training examples that are correctly classified (see, for example, [15]).

In our case, the SVM is used to classify strings of amino acids. The feature space, upon which the separating hyperplane is built, is defined by a mapping from each possible string $s$ to a high-dimensional vector $\phi(s)$. For example, in the case of the *blended*

*spectrum kernel* [15], each component $\phi_\alpha(s)$ is the frequency of occurrence in $s$ of a specific substring $\alpha$ that we call a *segment*. The whole vector $\phi(s)$ is the collection of all these frequencies for each possible segment of at most $p$ symbols. Consequently, vector $\phi(s)$ has $\sum_{i=1}^{p} |\Sigma|^i$ components for an alphabet $\Sigma$ containing $|\Sigma|$ symbols. If $w$ denotes the normal vector of the separating hyperplane, and $b$ its bias (which is related to the distance that the hyperplane has from the origin), then the output $f(s)$ of the SVM classifier, on input string $s$, is given by

$$f(s) \;=\; \mathrm{sgn}\left(\langle w, \phi(s)\rangle + b\right) \,,$$

where $\mathrm{sgn}(a) = +1$ if $a > 0$ and $-1$ otherwise, and where $\langle w, \phi(s)\rangle$ denotes the inner product between vectors $w$ and $\phi(s)$. We have $\langle w, \phi(s)\rangle = \sum_{i=1}^{d} w_i \phi_i(s)$ for $d$-dimensional vectors. The normal vector $w$ is often called the *discriminant* or the *weight vector*.

### 5.6.1 Learning in spaces of large dimensionality

Constructing a separating hyperplane in spaces of very large dimensionality has potentially two serious drawbacks. The first drawback concerns the obvious danger of *overfitting*. Indeed, with so many degrees of freedom for a vector $w$ having more components than the number of training examples, there may exist many different $w$ having a high probability of error while making very few training errors. However, several theoretical results [290, 15] indicate that overfitting is unlikely to occur when a large separating margin is found on the (numerous) correctly classified examples—thus giving theoretical support to the learning strategy used by the SVM.

The second potential drawback concerns the computational cost of using very high dimensional feature vectors $\phi(s_1), \phi(s_2), \ldots, \phi(s_m)$ of training examples. As we now demonstrate, this drawback can elegantly be avoided by using *kernels* instead of feature vectors. The basic idea consists of representing the discriminant $w$ as a linear combination of the feature vectors of the training examples. More precisely, given a training set $\{(s_1, y_1), (s_2, y_2), \ldots, (s_m, y_m)\}$ and a mapping $\phi(\cdot)$, we write $w = \sum_{i=1}^{m} \alpha_i y_i \phi(s_i)$. The set $\{\alpha_1, \ldots, \alpha_m\}$ is called the *dual representation* of the (primal) weight vector $w$. Consequently, the inner product $\langle w, \phi(s)\rangle$, used for computing the output of an SVM classifier, becomes

$$\langle w, \phi(s)\rangle = \sum_{i=1}^{m} \alpha_i y_i \langle \phi(s_i), \phi(s)\rangle \;=\; \sum_{i=1}^{m} \alpha_i y_i k(s_i, s) \,,$$

where $k(s,t) \stackrel{def}{=} \langle \phi(s), \phi(t) \rangle$ defines the *kernel function* associated with the feature map $\phi(\cdot)$. With the dual representation, the SVM classifier is entirely described in terms of the training examples $s_i$ having a non-zero value for $\alpha_i$. These examples are called *support vectors*. The so-called "kernel trick" consists of using $k(s,t)$ without explicitly computing $\langle \phi(s), \phi(t) \rangle$—a computationally prohibitive task for feature vectors of very large dimensionality. This is possible for many feature maps $\phi(\cdot)$. Consider again, for example, the *blended spectrum* (BS) kernel where each component $\phi_\alpha(s)$ is the frequency of occurrence of a segment $\alpha$ in string $s$ (for all words of at most $p$ characters of an alphabet $\Sigma$). In this case, instead of performing $\sum_{i=1}^{p} |\Sigma|^i$ multiplications to compute explicitly $\langle \phi(s), \phi(t) \rangle$, we can compute, for each position $i$ in string $s$ and each position $j$ in string $t$, the number of consecutive symbols that matches in $s$ and $t$. We use the big-Oh notation to provide an upper bound to the running time of algorithms. Let $T(n)$ denote the execution time of an algorithm on an input of size $n$. We say that $T(n)$ is in $O(g(n))$ if and only if there exists a constant $c$ and a critical $n_0$ such that $T(n) \leq cg(n)$ for all $n \geq n_0$. The blended spectrum kernel requires at most $O(p \cdot |s| \cdot |t|)$ time for each string pair $(s,t)$—an enormous improvement over the $\Omega(|\Sigma|^p)$ time required for the explicit computation of the inner product between a pair of feature vectors. In fact, there exists an algorithm [15] for computing the blended spectrum kernel in $O(p \cdot \max(|s|, |t|))$ time.

### 5.6.2 The distant segments kernel

The blended spectrum kernel is interesting because it contains all the information concerning the population of segments that are present in a string of symbols without considering their relative positions. Here, we propose the *distant segments* (DS) kernel that, in some sense, extends the BS kernel to include (relative) positional information of segments in a string of symbols.

If one considers the frequencies of all possible segment distances inside a string as its features, then a precise comparison can be done between any pair of strings. Remote protein homology can be detected using distances between polypeptide segments [23]. For any string $s$ of amino acids, these authors used explicitly a feature vector $\phi(s)$ where each component $\phi_{d,\alpha,\alpha'}(s)$ denotes the number of times the (polypeptide) segment $\alpha'$ is located at distance $d$ (in units of symbols) following the (polypeptide) segment $\alpha$. They have restricted themselves to the case where $\alpha$ and $\alpha'$ have the same length $p$, with $p \leq 3$. Since the distance $d$ is measured from the first symbol in $\alpha$ to the first symbol in $\alpha'$, the $d = 0$ components of $\phi(s)$, *i.e.*, $\phi_{0,\alpha,\alpha'}(s)$, are non-zero only for $\alpha = \alpha'$

and represent the number of occurrences of segment $\alpha$ in string $s$. Consequently, this feature vector strictly includes all the components of the feature vector associated with the BS kernel but is limited to segments of size $p$ (for $p \leq 3$). By working with the explicit feature vectors, these authors were able to obtain easily the components of the discriminant vector $w$ that are largest in magnitude and, consequently, are the most relevant for the binary classification task. However, the memory requirement of their algorithm increases exponentialy in $p$. Not surprisingly, only the results for $p \leq 3$ were reported by [23].

Despite these limitations, the results of [23] clearly show the relevance of having features representing the frequency of occurrences of pairs of segments that are separated by some distance for protein remote homology detection. Hence, we propose in this section the *distance segments* (DS) kernel that potentially includes all the features considered by [23] without limiting ourselves to $p \leq 3$ and to the case where the words (or segments) have to be of the same length. Indeed, we find no obvious biological motivation for these restrictions. Also, as we will show, there is no loss of interpretability of the results by using a kernel instead of the feature vectors. In particular, we can easily obtain the most significant components of the discriminant $w$ by using a kernel. We will show that the time and space required for computing the kernel matrix and obtaining the most significant components of the discriminant $w$ are bounded polynomially in terms of all the relevant parameters.

Consider a protein as a string of symbols from the alphabet $\Sigma$ of amino acids. $\Sigma^*$ represents the set of all finite strings (including the empty string). For $\mu \in \Sigma^*$, $|\mu|$ denotes the length of the string $\mu$. Throughout the paper, $s$, $t$, $\alpha$, $\mu$ and $\nu$ will denote strings of $\Sigma^*$, whereas $\theta$ and $\delta$ will be lengths of such strings. Moreover, $\mu\nu$ will denote the concatenation of $\mu$ and $\nu$. The DS kernel is based on the following set. Given a string $s$, let $\mathcal{S}^{\delta}_{\alpha,\alpha'}(s)$ be the set of all the occurrences of substrings of length $\delta$ that are beginning by segment $\alpha$ and ending by segment $\alpha'$. More precisely,

$$\mathcal{S}^{\delta}_{\alpha,\alpha'}(s) \stackrel{def}{=} \{(\mu,\alpha,\nu,\alpha',\mu') : \; s=\mu\alpha\nu\alpha'\mu' \;\; \wedge \;\; 1\leq|\alpha| \;\; \wedge 1\leq|\alpha'| \;\; \wedge \; 0 \leq |\nu| \;\; \wedge \;\; \delta=|s|-|\mu|-|\mu'|\} \; . \tag{5.1}$$

Note that the substring length $\delta$ is related to the distance $d$ of [23] by $\delta = d+|\alpha'|$ where $d = |\alpha| + |\nu|$ when $|\alpha|$ and $|\alpha'|$ do not overlap. Note also that, in contrast with [23], we may have $|\alpha| \neq |\alpha'|$. Moreover, the segments $\alpha$ and $\alpha'$ never overlap since $\mu\alpha\nu\alpha'\mu'$ equals to the whole string $s$ and $0 \leq |\nu|$. We have made this choice because it appeared biologically more plausible to have a distance ranging from the end of the first segment

to the beginning of the second segment. Nevertheless, we will see shortly that we can include the possibility of overlap between segments with a very minor modification of the kernel.

The DS kernel is defined by the following inner product

$$k_{DS}^{\delta_m,\theta_m}(s,t) \stackrel{def}{=} \langle \phi_{DS}^{\delta_m,\theta_m}(s), \phi_{DS}^{\delta_m,\theta_m}(t) \rangle \,, \tag{5.2}$$

where $\phi_{DS}^{\delta_m,\theta_m}(s)$ is the feature vector

$$\phi_{DS}^{\delta_m,\theta_m}(s) \stackrel{def}{=} \left( \left| \mathcal{S}_{\alpha,\alpha'}^{\delta}(s) \right| \right)_{\{(\delta,\alpha,\alpha') :\, 1 \leq |\alpha| \leq \theta_m \,\wedge\, 1 \leq |\alpha'| \leq \theta_m \,\wedge\, |\alpha|+|\alpha'| \leq \delta \leq \delta_m\}} \,.$$

Hence, the kernel is computed for a fixed maximum value $\theta_m$ of segment sizes and a fixed maximum value $\delta_m$ of substring length. Note that, the number of strings of size $\theta$ of $\Sigma^*$ grows exponentially with respect to $\theta$. Fortunately, we are able to avoid this potentially devastating combinatorial explosion in our computation of $k_{DS}^{\delta_m,\theta_m}(s,t)$. Figure 5.1 shows the pseudo-code of the algorithm. In the pseudo-code, $s[i]$ denotes the symbol located at position $i$ in the string $s$ (with $i \in \{1, 2, \ldots, |s|\}$). Moreover, for any integers $i, j$, $\binom{j}{i}$ denotes $\frac{j!}{i!\,(j-i)!}$ if $0 \leq i \leq j$, and 0 otherwise. Admittedly, it is certainly not clear that the algorithm of Figure 5.1 actually computes the value of $k_{DS}^{\delta_m,\theta_m}(s,t)$ given by Equation 5.2. Hence, a proof of correctness of this algorithm is presented at the appendix (located after the conclusion). The worst-case running time is easy to obtain because the algorithm is essentially composed of three imbricated loops: one for $j_s \in \{0, \ldots, |s|-1\}$, one for $j_t \in \{0, \ldots, |t|-1\}$, and one for $i \in \{1, \ldots, \min(|s|, |t|, \delta_m)\}$. The time complexity is therefore in $O(|s| \cdot |t| \cdot \min(|s|, |t|, \delta_m))$.

Note that the definition of the DS-kernel can be easily modified in order to accept overlaps between $\alpha$ and $\alpha'$. Indeed, when overlaps are permitted, they can only occur when both $\alpha$ and $\alpha'$ start and end in $\{j_s+i_0, \ldots, j_s+i_1-1\}$. The number of elements of $\mathcal{S}_{(j_s,j_t)}$ for which $i_{2r} \leq \delta < i_{2r+1}$ is thus the same for all values of $r$, including $r = 0$. Consequently, the algorithm to compute the DS kernel, when overlaps are permitted, is the same as the one in Figure 5.1 except that we need the replace the last two lines of the FOR loop, involved in the computation of $c$, by the single line:

$$c \leftarrow c \; + \min(\theta_m, i_1 - i_0) \cdot \sum_{r=0}^{k} \left( \binom{l_r}{2} - \binom{l_r - \theta_m}{2} \right) .$$

Similar simple modifications can be performed for the more restrictive case of $|\alpha| = |\alpha'|$.

### 5.6.3 Extracting the discriminant vector with the distant segments kernel

We now show how to extract (with reasonable time and space resources) the components of the discriminant $w$ that are non-zero. Recall that $w = \sum_{i=1}^{l} \alpha_i y_i \phi(s_i)$ when the SVM contains $l$ support vectors $\{(s_1, y_1), \ldots, (s_l, y_l)\}$. Recall also that each feature $\phi_{\delta,\alpha,\alpha'}(s_i)$ is identified by a triplet $(\delta, \alpha, \alpha')$, with $\delta \geq |\alpha| + |\alpha'|$. Hence, to obtain the non-zero valued components of $w$, we first obtain the non-zero valued features $\phi_{\delta,\alpha,\alpha'}(s_i)$ from each support vector (with Algorithm EXTRACT-FEATURES of Figure 5.2) and then collect and merge every feature of each support vector by multiplying each of them by $\alpha_i y_i$ (with Algorithm EXTRACT-DISCRIMINANT of Figure 5.3).

We transform each support vector $\phi(s_i)$ into a *Map* of features. Each *Map key* is an identifier for a $(\delta, \alpha, \alpha')$ having $\phi_{\delta,\alpha,\alpha'}(s_i) > 0$. The *Map value* is given by $\phi_{\delta,\alpha,\alpha'}(s_i)$ for each key.

The worst-case access time for an AVL-tree-Map of $n$ elements is $O(\log n)$. Hence, from Figure 5.2, the time complexity of extracting all the (non-zero valued) features of a support vector is in $O(|s|\theta_m^2\delta_m \cdot \log(|s|\theta_m^2\delta_m))$. Moreover, since the total number of features inserted to the Map by the algorithm EXTRACT-DISCRIMINANT is at most $l \cdot |s| \cdot \theta_m^2\delta_m$, the time complexity of extracting all the non-zero valued components of $w$ is in $O(l|s|\theta_m^2\delta_m \cdot \log(l|s|\theta_m^2\delta_m))$.

### 5.6.4 SVM

We have used a publicly available SVM software, named SVM$^{\text{light}}$ [132], for predicting the coreceptor usage. Learning SVM classifier requires to choose the right trade-off between training accuracy and the magnitude of the separating margin on the correctly classified examples. This trade-off is generally captured by a so-called soft-margin hyperparameter $C$.

The learner must choose the value of $C$ from the training set only—the testing set must be used only for estimating the performance of the final classifier. We have used the (well-known) 10-fold cross-validation method (on the training set) to determine the best value of $C$ and the best values of the kernel hyperparameters (that we describe below). Once the values of all the hyperparameters were found, we used these values to train the final SVM classifier on the whole training set.

### 5.6.5 Selected metrics

The testing of the final SVM classifier was done according to several metrics. Let P and N denote, respectively, the number of positive examples and the number of negative examples in the test set. Let TP, the number of "true positives", denote the number of positive testing examples that are classified (by the SVM) as positive. A similar definition applies to TN, the number of "true negatives". Let FP, the number of "false positives", denote the number of negative testing examples that are classified as positive. A similar definition applied to FN, the number of "false negatives". To quantify the "fitness" of the final SVM classifier, we have computed the *accuracy*, which is (TP+TN)/(P+T), the *sensitivity*, which is TP/P, and the specificity, which is TN/N. Finally, for those who cannot decide how much to weight the cost of a false positive, in comparison with a false negative, we have computed the "area under the ROC curve" as described by [25].

Unlike the other metrics, the accuracy (which is 1 - the testing error) has the advantage of having very tight confidence intervals that can be computed straightforwardly from the binomial tail inversion, as described by [146]. We have used this method to find if whether or not the observed difference of testing accuracy (between two classifiers) was statistically significant. We have reported the results only when a statistically significant difference was observed with a 90% confidence level.

### 5.6.6 Selected string kernels

One of the kernel used was the blended spectrum (BS) kernel that we have described above. Recall that the feature space, for this kernel, is the count of all $k$-mers with $1 \leq k \leq p$. Hence $p$ is the sole hyperparameter of this kernel.

We have also used the local alignment (LA) kernel [255] which can be thought of as a soft-max version of the Smith-Waterman local alignment algorithm for pair of sequences. Indeed, instead of considering the alignment that maximizes the Smith-Waterman (SW) score, the LA kernel considers every local alignment with a Gibbs distribution that depends on its SW score. Unfortunately, the LA kernel has too many hyperparameters precluding their optimization by cross-validation. Hence, a number of choices were made based on the results of [255]. Namely, the alignment parameters were set to (BLOSUM 62, $e = 11$, $d = 1$) and the empirical kernel map of the LA kernel was used. The hyperparameter $\beta$ was the only one that was adjusted by cross-validation.

Of course, the proposed distant segments (DS) kernel was also tested. The $\theta_m$ hyperparameter was set to $\delta_m$ to avoid the limitation of segment length. Hence, $\delta_m$ was the sole hyperparameter for this kernel that was optimized by cross-validation.

Other interesting kernels, not considered here because they yielded inferior results (according to [255], and [23]) on the remote protein homology detection problem, include the mismatch kernel [155] and the pairwise kernel [28].

### 5.6.7 Datasets

The V3 loop sequence and coreceptor usage of HIV-1 samples were retrieved from Los Alamos National Laboratory HIV Databases (http://www.hiv.lanl.gov/) through available online forms.

Every sample had a unique GENBANK identifier. Sequences containing #, $ or * were eliminated from the dataset. The signification of these symbols was reported by Brian Foley of Los Alamos National Laboratory (personal communication). The # character indicates that the codon could not be translated, either because it had a gap character in it (a frame-shifting deletion in the virus RNA), or an ambiguity code (such as R for purine). The $ and * symbols represent a stop codon in the RNA sequence. TAA, TGA or TAG are stop codons. The dataset was first shuffled and then splitted half-half, yielding a training and a testing set. The decision to shuffle the dataset was made to increase the probability that both the training and testing examples are obtained from the same distribution. The decision to use half of the dataset for testing was made in order to obtain tight confidence intervals for accuracy.

Samples having the same V3 loop sequence and a different coreceptor usage label are called *contradictions*. Contradictions were kept in the datasets to have prediction performances that take into account the biological reality of dual tropism for which frontiers are not well defined.

Statistics were compiled for the coreceptor usage distribution, the count of contradictions, the amount of samples in each clades and the distribution of the V3 loop length.

## 5.7 Results

Here we report statistics on our datasets, namely the distribution, contradictions, subtypes and the varying lengths. We also show the results of our classifiers on the HIV-1

coreceptor usage prediction task, a brief summary of existing methods and an analysis of the discriminant vector with the distant segments kernel.

### 5.7.1 Statistics

In Table 5.1 is reported the distribution of coreceptor usages in the datasets created from Los Alamos National Laboratory HIV Databases data. In the training set, there are 1225 CCR5-utilizing samples (85.9%), 375 CXCR4-utilizing samples (26.3%) and 175 CCR5-and-CXCR4-utilizing samples (12.2%). The distribution is approximatly the same in the test set. There are contradictions (entries with the same V3 sequence and a different coreceptor usage) in all classes of our datasets. A majority of viruses can use CCR5 in our datasets.

In Table 5.2, the count is reported regarding HIV-1 subtypes, also known as genetic clades. HIV-1 subtype B is the most numerous in our datasets. The clade information is not an attribute that we provided to our classifiers, we only built our method on the primary structure of the V3 loop. Therefore, our method is independant of the clades.

The V3 loops have variable lengths among the virions of a population. In our dataset (Table 5.3), the majority of sequences has exactly 36 residues, although the length ranges from 31 to 40.

### 5.7.2 Coreceptor usage predictions

Classification results on the three different tasks (CCR5, CXCR4, CCR5-and-CXCR4) are presented in Table 5.4 for three different kernels .

For the CCR5-usage prediction task, the SVM classifier achieved a testing accuracy of 96.63%, 96.42%, and 96.35%, respectively, for the BS, LA, and DS kernels. By using the binomial tail inversion method of [146], we find no statistically significant difference, with 90% confidence, between kernels.

For the CXCR4-usage prediction task, the SVM classifier achieved a testing accuracy of 93.68%, 92.21%, and 94.80%, respectively, for the BS, LA, and DS kernels. By using the binomial tail inversion method of [146], we find that the difference is statistically significant, with 90% confidence, for the DS versus the LA kernel.

For the CCR5-and-CXCR4-usage task, the SVM classifier achieved a testing accuracy of 94.38%, 92.28 %, and 95.15%, respectively, for the BS, LA, and DS kernels. Again,

we find that the difference is statistically significant, with 90% confidence, for the DS versus the LA kernel.

Overall, all the tested string kernels perform well on the CCR5 task, but the DS kernel is significantly better than the LA kernel (with 90% confidence) for the CXCR4 and CCR5-and-CXCR4 tasks. For these two prediction tasks, the performance of the BS kernel was closer to the one obtained for the DS kernel than the one obtained for the LA kernel.

### 5.7.3 Classification with the perfect deterministic classifier

Also present in Table 5.4 are the results of the *perfect deterministic classifier*. This classifier is the deterministic classifier achieving the highest possible accuracy on the test set. For any input string $s$ in a testing set $T$, the perfect determinist classifier ($h^*$) returns the most frequently encountered class label for string $s$ in $T$. Hence, the accuracy on $T$ of $h^*$ is an overall measure of the amount of contradictions that are present in $T$. There are no contradictions in $T$ if and only if the testing accuracy of $h^*$ is 100%. As shown in Table 5.4, there is a significant amount of contradictions in the test set $T$. These results indicate that any deterministic classifier cannot achieve an accuracy greater than 99.15%, 98.66% and 97.96%, respectively for the CCR5, CXCR4, and CCR5-and-CXCR4 coreceptor usage tasks.

### 5.7.4 Discriminative power

To determine if a SVM classifier equipped with the distant segments (DS) kernel had enough discriminative power to achieve the accuracy of perfect determinist classifier, we trained the SVM, equipped with the DS kernel, *on the testing set*. From the results of Table 5.4, we conclude that the SVM equipped with the DS kernel possess sufficient discriminative power since it achieved (almost) the same accuracy as the perfect deterministic classifier for all three tasks. Hence, the fact that the SVM with the DS kernel does not achieve the same accuracy as the perfect determinist classifier when it is obtained from the training set (as indicated in Table 5.4) is not due to a lack of discriminative power from the part of the learner.

### 5.7.5 Discriminant vectors

The discriminant vector that maximizes the soft-margin has (almost always) many non-zero valued components which can be extracted by the algorithm of Figure 5.3.

We examine which components of the discriminant vector have the largest absolute magnitude. These components give weight to the most relevant features for a given classification task. In Figure 5.4, we describe the most relevant features for each tasks. Only the 20 most significant features are shown.

A subset of positive-weighted features shown for CCR5-utilizing viruses are also in the negative-weighted features shown for CXCR4-utilizing viruses. Furthermore, a subset of positive-weighted features shown for CXCR4-utilizing viruses are also in the negative-weighted features reported for CCR5-utilizing viruses. Thus, CCR5 and CXCR4 discriminant models are complementary. However, since 3 tropisms exist (R5, X4 and R5X4), features contributing to CCR5-and-CXCR4 should also include some of the features contributing to CCR5 and some of the features contributing to CXCR4. Among shown positive-weighted features for CCR5-and-CXCR4, there are features that also contribute to CXCR4 ([8,R,R], [13,R,T], [9,R,R]). On another hand, this is not the case for CCR5. However, only the twenty most relevant features have been shown and there are many more features, with similar weights, that contribute to the discriminant vector. In fact, the classifiers that we have obtained depend on a very large number of features (instead of a very small subset of relevant features).

## 5.8 Discussion

The proposed HIV-1 coreceptor-usage prediction tool achieved very high accuracy in comparison with other existing prediction methods. In view of the results of Pillai et al, we have shown that the SVM classification accuracy can be greatly improved with the usage of a string kernel. Surprisingly, the local alignment (LA) kernel, which makes an explicit use of biologically-motivated scoring matrices (such as BLOSUM 62), turns out to be outperformed by the blended spectrum (BS) and the distant segments (DS) kernels which do not try to exploit any concept of similarity between residues but rely, instead, on a very large set of easily-interpretable features. Thus, a weighted-majority vote over a very high number of simple features constitutes a very productive approach, that is both sensitive and specific to what it is trained for, and applies well in the field of viral phenotype prediction.

### 5.8.1 Comparison with available bioinformatic methods

In Table 5.5, we show a summary of the available methods. The simplest method (the charge rule) has an accuracy of 87.45%. Thus, the charge rule is the worst method

presented in table 5.5. The SVM with string kernels is the only approach without multiple alignments. Therefore, V3 sequences with many indels can be used with our method, but not with the other. These other methods were not directly tested here with our datasets because they all rely on multiple alignments. The purpose of those alignments is to produce a consensus and to yield transformed sequences having all the same length. As indicated by the size of the training set in those methods, sequences having larger indels were discarded, thus making these datasets smaller. Most of the methods rely on cross-validation to perform quality assessment but, as we have mentioned, this is problematic when multiple alignments are performed prior to learning, since, in these cases, the testing set in each fold is used for the construction of the tested classifier. It is also important to mention that the various methods presented in Table 5.5 do not produce predictors for the same coreceptor usage task. Indeed, the definition of X4 viruses is not always the same: some authors refer to it as CXCR4-only while other use it as CXCR4-utilizing. It is thus unfeasible to assess the fitness of these approaches, which are twisted by cross-validation, multiple alignments and heterogeneous dataset composition.

The work by Lamers and colleagues [12] is the first development in HIV-1 coreceptor usage prediction regarding dual-tropic viruses. Using evolved neural networks, an accuracy of 75.50% was achieved on a training set of 149 sequences with the cross-validation method. However, the SVM equipped with the distant segments kernel reached an accuracy of 95.15% on a large test set (1425 sequences) in our experiments. Thus, our SVM outperforms the neural network described by Lamers and colleagues [12] for the prediction of dual-tropic viruses.

### 5.8.2 Los Alamos National Laboratory HIV Databases

Although we used only the Los Alamos National Laboratory HIV Databases as our source of sequence information, it is notable that this data provider represents a meta-resource, fetching bioinformation from databases around the planet, namely GenBank (USA, http://www.ncbi.nlm.nih.gov/Genbank/), EMBL (Europe, http://www.ebi.ac.uk/embl/) and DDBJ (Japan, http://www.ddbj.nig.ac.jp/). Researchers cannot directly send their HIV sequences to LANL, but it is clear that this approach makes this database less likely to contain errors.

### 5.8.3 Noise

The primary cause of contradictions (e.g. a sequence having two or more phenotypes) remains uncharacterized. It may be due to a particular mix, to some extent, of virion envelope attributes (regions other than the V3) and of the host cell receptor counterparts. As genotypic assays, based on bioinformatics prediction software, rely on sequencing technologies, they are likely to play a more important role in clinical settings as sequencing cost drops. Next-generation sequencing platforms promise a radical improvement on the throughput and more affordable prices. Meanwhile, effective algorithmic methods with proven statistical significance must be developed. Bioinformatics practitioners have to innovate by creating new algorithms to deal with large datasets and need to take into consideration sequencing errors and noise in phenotypic assay readouts. Consequently, we investigated the use of statistical machine learning algorithms, such as the SVM, whose robustness against noise has been observed in many classification tasks. The high accuracy results we have obtained here indicate that this is also the case for the task of predicting the coreceptor usage of HIV-1. While it remains uncertain whether or not other components of the HIV-1 envelope contribute to the predictability of the viral phenotype, we have shown that the V3 loop alone produces very acceptable outputs despite the presence of a small amount of noise.

### 5.8.4 Web server

To allow HIV researchers to test our method on the web, we have implemented a web server for the HIV-1 coreceptor usage prediction. The address of this web server is http://genome.ulaval.ca/hiv-dskernel. In this setting, one has to submit fasta-formatted V3 sequences in a web form. Then, using the dual representation of the SVM with the distant segments kernel, the software predicts the coreceptor usage of each submitted viral sequence. Those predictions are characterized by high accuracy (according to results in Table 5.4). Source codes for the web server and for a command-line back-end are available in additional file 1.

## 5.9 Conclusions

To our knowledge, this is the first paper that investigates the use of string kernels for predicting the coreceptor usage of HIV-1. Our contributions include a novel string kernel (the distant segments kernel), a SVM predictor for HIV-1 coreceptor usage with the identification of the most relevant features and state-of-the-art results on accuracy,

specificity, sensitivity and receiver operating characteristic. As suggested, string kernels outperform all published algorithms for HIV-1 coreceptor usage prediction. Large margin classifiers and string kernels promise improvements in drug selection, namely CCR5 coreceptor inhibitors and CXCR4 coreceptor inhibitors, in clinical settings. Since the binding of an envelope protein to a receptor/coreceptor prior to infection is not specific to HIV-1, one could extend this work to other diseases. Furthermore, most ligand interactions could be analyzed in such a fashion. Detailed features in primary structures (DNA or protein sequences) can be elucidated with the proposed bioinformatic method.

Although we have exposed that even the perfect algorithm (entitled "Perfect determinist classifier") can not reach faultless outcomes, we have also empirically demonstrated that our algorithms are very competitive (more than 96% with distant segments for CCR5). It is thus feasible to apply kernel methods based on features in primary structures to compare sequence information in the perspective of predicting a phenotype. The distant segments kernel has broad applicability in HIV research such as drug resistance, coreceptor usage (as shown in this paper), immune escape, and other viral phenotypes.

## 5.10 Appendix

### 5.10.1 Proof of the correctness of the distant segments kernel

We now prove that the algorithm DISTANT-SEGMENTS-KERNEL$(s, t, \delta_m, \theta_m)$ does, indeed, compute $k_{DS}^{\delta_m,\theta_m}(s,t)$ as defined by Equation 5.2.

*Proof.* For each pair $(j_s, j_t)$ such that $s[j_s+1] = t[j_t+1]$ and each $\delta \geq 2$, let us define $\mathcal{S}_{(j_s,j_t)}$ to be the set of all triples $\Big(\delta,\ (\mu_s, \alpha, \nu_s, \alpha', \mu'_s),\ (\mu_t, \alpha, \nu_t, \alpha', \mu'_t)\Big)$ such that

- $\delta \leq \delta_m$ ;
- $|\alpha| \leq \theta_m$ and $|\alpha'| \leq \theta_m$;
- $(\mu_u, \alpha, \nu_u, \alpha', \mu'_u) \in \mathcal{S}^{\delta}_{\alpha,\alpha'}(u)$ for $u = s$ and for $u = t$;
- $|\mu_s| = j_s$ and $|\mu_t| = j_t$.

Clearly, $k_{DS}^{\delta_m,\theta_m}(s,t)$ is the sum of all the values of $|\mathcal{S}_{(j_s,j_t)}|$ over all the possible pairs $(j_s, j_t)$. Moreover, it is easy to see that $|\mathcal{S}_{(j_s,j_t)}|$ can be computed only from the knowledge of the set of indices $i \in \{1, \ldots, n\}$ satisfying property $P(i) := s[j_s+i] = t[j_t+i]$.

Note that when the test of the first WHILE loop is performed, the value of $i$ is such that $P(i)$ is valid but not $P(i-1)$. Moreover, in the second WHILE loop, $P(i)$ remains valid, except for the last test. Thus, $s[j_s+i]=t[j_t+i]$ if and only if $i_{2r} \leq i < i_{2r+1}$ for some $r \in \{0,\ldots,k\}$. This, in turn, implies that each element of $\mathcal{S}_{(j_s,j_t)}$ must be such that $i_{2r} \leq \delta < i_{2r+1}$ for some $r$. To obtain the result, it is therefore sufficient to prove that both of theses properties hold.

**P**1 The number of elements of $\mathcal{S}_{(j_s,j_t)}$, for which $i_0 < \delta < i_1$, is given by

$$\binom{l_0}{3} - 2\binom{l_0-\theta_m}{3} + \binom{l_0-2\theta_m}{3}.$$

**P**2 For $r \in \{1,\ldots,k\}$, the number of elements of $\mathcal{S}_{(j_s,j_t)}$, for which $i_{2r} \leq \delta < i_{2r+1}$, is given by

$$\min(\theta_m, i_1-i_0) \cdot \left(\binom{l_r}{2} - \binom{l_r-\theta_m}{2}\right).$$

To prove these properties, we will use the fact that, if $1 \leq i \leq k$, then $\binom{k}{i}$ counts the number of sequences $\langle a_1,\ldots,a_i\rangle$ satisfying $1 \leq a_1 < a_2 < \cdots < a_i \leq k$ where $\{a_1,a_2,...,a_i\}$ are $i$ string positions. Moreover, for any substring $u$ of $s$, let us denote by $b_u$ the starting position of $u$ in $s$, and by $e_u$ the first position of $s$ after the substring $u$ (if $s$ ends with $u$, choose $e_u=|s|+1$).

We first prove **P2**. Fix an $r$. Since $i_0=1$, we have that any $s$-substring $\alpha$ of length $\leq \theta_m$ with $b_\alpha = j_s+i_0$ and $e_\alpha \leq j_s+i_1$ together with any $s$-substring $\alpha'$ of length $\leq \theta_m$ such that

$$j_s+i_{2r} \leq b_{\alpha'} < e_{\alpha'} \leq j_s+i_{2r+1},$$

will give rise to exactly one element of $\mathcal{S}_{(j_s,j_t)}$ with $i_{2r} \leq \delta < i_{2r+1}$. Conversely, each element $\mathcal{S}_{(j_s,j_t)}$ such that $i_{2r} \leq \delta < i_{2r+1}$ will have an $\alpha$ and an $\alpha'$ with these properties. Since $\alpha$ has to start at $j_s+i_0$, it is easy to see that the number of such possible $\alpha$ is exactly $\min(\theta_m,\, i_1-i_0)$. Thus let us show that the number of possible $\alpha'$ is exactly $\binom{l_r}{2} - \binom{l_r-\theta_m}{2}$.

Since $l_r$ gives the number of positions from $j_s+i_{2r}$ to $j_s+i_{2r+1}$ inclusively, $\binom{l_r}{2}$ counts all the possible choices of $b_{\alpha'}$ and $e_{\alpha'}$ with $j_s+i_{2r} \leq b_{\alpha'} < e_{\alpha'} \leq j_s+i_{2r+1}$. Thus $\binom{l_r}{2}$ counts the number of possible strings $\alpha'$ of all possible lengths (including lengths $> \theta_m$). On another hand, the number of $\alpha'$ having a length $> \theta_m$ is equal to $\binom{l_r-\theta_m}{2}$. Indeed, if $l_r-\theta_m<2$, $\binom{l_r-\theta_m}{2}=0$ as wanted, and otherwise, there is a one-to-one correspondence

between the set of all sequences $\langle a_1, a_2\rangle$ such that $1 \leq a_1 < a_2 \leq l_r - \theta_m$ and the set of all $\alpha'$ of length $> \theta_m$. The correspondence is obtained by putting $b_{\alpha'} = i_{2r} + a_1 - 1$ and $e_{\alpha'} = i_{2r} + \theta_m + a_2 - 1$.

The proof for **P1** is similar to the one for **P2** except that we have to consider the fact that both $\alpha$ and $\alpha'$ start and end in $\{j_s+i_0, \ldots, j_s+i_1-1\}$. Since no overlap is allowed and $b_\alpha = j_s + i_0$, we must have

$$j_s + i_0 \leq e_\alpha - 1 < b_{\alpha'} < e_{\alpha'} \leq j_s + i_1 \,.$$

Since $l_0$ gives the number of positions from $j_s+i_0$ to $j_s+i_1$ inclusively, $\binom{l_0}{3}$ counts all the possible choices of $\alpha$ and $\alpha'$ for all possible lengths. Recall that if $l_0 < 3$, which can only occur if $i_1 = i_0 + 1$, we have that $\binom{l_0}{3} = 0$, as wanted.

On another hand, $\binom{l_0-\theta_m}{3}$ counts all the possible choices of $\alpha$ of length $> \theta_m$ and of $\alpha'$ of arbitrary length. This set of possible choices is non empty only if $l_0 - \theta_m \geq 3$ and, then, the one-to-one correspondence between a sequence $\langle a_1, a_2, a_3\rangle$ such that $1 \leq a_1 < a_2 < a_3 \leq l_0 - \theta_m$ and the values of $\langle e_\alpha, b_{\alpha'}, e_{\alpha'}\rangle$ is

$$e_\alpha - 1 = j_s + \theta_m + a_1, \quad b_{\alpha'} = j_s + \theta_m + a_2 \quad \text{and} \quad e_{\alpha'} = j_s + \theta_m + a_3.$$

Similarly, $\binom{l_0-\theta_m}{3}$ counts all the possible choices of $\alpha'$ of length $> \theta_m$ and of $\alpha$ of arbitrary length, the correspondence being $e_\alpha - 1 = j_s + a_1$, $b_{\alpha'} = j_s + a_2$ and $e_{\alpha'} = j_s + \theta_m + a_3$. Finally, $\binom{l_0-2\theta_m}{3}$ counts all the possible choices of $\alpha$ and $\alpha'$, both of length $> \theta_m$. In the cases where such possible choices exist (i.e., if $l_0 - 2\theta_m \geq 3$), the correspondence is $e_\alpha - 1 = j_s + \theta_m + a_1$, $b_{\alpha'} = j_s + \theta_m + a_2$ and $e_{\alpha'} = j_s + 2\theta_m + a_3$. Then, property **P1** immediately follows from the inclusion-exclusion argument. □

## 5.11 Competing interests

The authors declare that they have no competing interests.

## 5.12 Authors' contributions

SB,MM,FL and JC drafted the manuscript. FL wrote the proof for the distant segments kernel. SB performed experiments. SB,MM,FL and JC approved the manuscript.

## 5.13 Acknowledgements

This project was funded by the Canadian Institutes of Health Research and by the Natural Sciences and Engineering Research Council of Canada (MM, 122405 and FL, 262067). JC is the holder of Canada Research Chair in Medical Genomics.

## 5.14 References

[1] Pillai S, Good B, Richman D, Corbeil J: **A new perspective on V3 phenotype prediction**. *AIDS Res. Hum. Retroviruses* 2003, **19**:145–149.

[2] Richman D, Bozzette S: **The impact of the syncytium-inducing phenotype of human immunodeficiency virus on disease progression**. *J. Infect. Dis.* 1994, **169**:968–974.

[3] Zhang L, Robertson P, Holmes EC, Cleland A, Leigh Brown A, Simmonds P: **Selection for specific V3 sequences on transmission of human immunodeficiency virus**. *J. Virol.* 1993, **67**:3345–56.

[4] Sirois M, Robitaille L, Sasik R, Estaquier J, Fortin J, Corbeil J: **R5 and X4 HIV viruses differentially modulate host gene expression in resting CD4+ T cells**. *AIDS Res. Hum. Retroviruses* 2008, **24**:485–493.

[5] Milich L, Margolin B, Swanstrom R: **V3 loop of the human immunodeficiency virus type 1 Env protein: interpreting sequence variability**. *J. Virol.* 1993, **67**:5623–5634.

[6] Fouchier R, Groenink M, Kootstra N, Tersmette M, Huisman H, Miedema F, Schuitemaker H: **Phenotype-associated sequence variation in the third variable domain of the human immunodeficiency virus type 1 gp120 molecule**. *J. Virol.* 1992, **66**:3183–3187.

[7] Resch W, Hoffman N, Swanstrom R: **Improved success of phenotype prediction of the human immunodeficiency virus type 1 from envelope variable loop 3 sequence using neural networks**. *Virology* 2001, **288**:51–62.

[8] Jensen M, Li F, van 't Wout A, Nickle D, Shriner D, He H, McLaughlin S, Shankarappa R, Margolick J, Mullins J: **Improved coreceptor usage prediction and genotypic monitoring of R5-to-X4 transition by motif analysis**

**of human immunodeficiency virus type 1 env V3 loop sequences**. *J. Virol.* 2003, **77**:13376–13388.

[9] Jensen M, Coetzer M, van 't Wout A, Morris L, Mullins J: **A reliable phenotype predictor for human immunodeficiency virus type 1 subtype C based on envelope V3 sequences**. *J. Virol.* 2006, **80**:4698–4704.

[10] Sander O, Sing T, Sommer I, Low A, Cheung P, Harrigan P, Lengauer T, Domingues F: **Structural descriptors of gp120 V3 loop for the prediction of HIV-1 coreceptor usage**. *PLoS Comput. Biol.* 2007, **3**:e58.

[11] Xu S, Huang X, Xu H, Zhang C: **Improved prediction of coreceptor usage and phenotype of HIV-1 based on combined features of V3 loop sequence using random forest**. *J. Microbiol.* 2007, **45**:441–446.

[12] Lamers S, Salemi M, McGrath M, Fogel G: **Prediction of R5, X4, and R5X4 HIV-1 coreceptor usage with evolved neural networks**. *IEEE/ACM Trans Comput Biol Bioinform* 2008, **5**:291–300.

[13] Lengauer T, Sander O, Sierra S, Thielen A, Kaiser R: **Bioinformatics prediction of HIV coreceptor usage**. *Nat. Biotechnol.* 2007, **25**:1407–1410.

[14] Cortes C, Vapnik V: **Support-Vector Networks**. *Machine Learning* 1995, **20**:273–297.

[15] Shawe-Taylor J, Cristianini N: *Kernel Methods for Pattern Analysis*. Cambridge University Press 2004.

[16] Saigo H, Vert J, Ueda N, Akutsu T: **Protein homology detection using string alignment kernels**. *Bioinformatics* 2004, **20**:1682–1689.

[17] Leslie C, Eskin E, Noble W: **The spectrum kernel: a string kernel for SVM protein classification**. *Pac Symp Biocomput* 2002, :564–575.

[18] Mefford M, Gorry P, Kunstman K, Wolinsky S, Gabuzda D: **Bioinformatic prediction programs underestimate the frequency of CXCR4 usage by R5X4 HIV type 1 in brain and other tissues**. *AIDS Res. Hum. Retroviruses* 2008, **24**:1215–1220.

[19] Raymond S, Delobel P, Mavigner M, Cazabat M, Souyris C, Sandres-Sauné K, Cuzin L, Marchou B, Massip P, Izopet J: **Correlation between genotypic predictions based on V3 sequences and phenotypic determination of HIV-1 tropism**. *AIDS* 2008, **22**:F11–16.

[20] Skrabal K, Low A, Dong W, Sing T, Cheung P, Mammano F, Harrigan P: **Determining human immunodeficiency virus coreceptor use in a clinical setting: degree of correlation between two phenotypic assays and a bioinformatic model**. *J. Clin. Microbiol.* 2007, **45**:279–284.

[21] Sing T, Low A, Beerenwinkel N, Sander O, Cheung P, Domingues F, Büch J, Däumer M, Kaiser R, Lengauer T, Harrigan P: **Predicting HIV coreceptor usage on the basis of genetic and clinical covariates**. *Antivir. Ther. (Lond.)* 2007, **12**:1097–1106.

[22] Vapnik V: *Statistical learning Theory*. New York: Wiley 1998.

[23] Lingner T, Meinicke P: **Remote homology detection based on oligomer distances**. *Bioinformatics* 2006, **22**:2224–2231.

[24] Joachims T: **Making large-Scale SVM Learning Practical**. In *Advances in Kernel Methods - Support Vector Learning*. Edited by Scholkopf B, Burges C, Smola A, MIT Press 1999.

[25] Gribskov M, Robinson N: **Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching**. *Comput. Chem.* 1996, **20**:25–33.

[26] Langford J: **Tutorial on practical prediction theory for classification**. *Journal of Machine Learning Research* 2005, **6**:273–306.

[27] Leslie C, Eskin E, Cohen A, Weston J, Noble W: **Mismatch string kernels for discriminative protein classification**. *Bioinformatics* 2004, **20**:467–476.

[28] Liao L, Noble W: **Combining pairwise sequence similarity and support vector machines for remote protein homology detection**. In *Proceedings of the Sixth Annual Conference on Research in Computational Molecular Biology* 2002:225–232.

## 5.15 Tables and captions

Table 5.1: Datasets. Contradictions are in parentheses.

| Coreceptor usage | Training set | | | Test set | | |
|---|---|---|---|---|---|---|
| | Negative examples | Positive examples | Total | Negative examples | Positive examples | Total |
| CCR5 | 200 (13) | 1225 (12) | 1425 (25) | 225 (22) | 1200 (16) | 1425 (38) |
| CXCR4 | 1050 (44) | 375 (18) | 1425 (62) | 1027 (28) | 398 (21) | 1425 (38) |
| CCR5 and CXCR4 | 1250 (57) | 175 (30) | 1425 (87) | 1252 (48) | 173 (35) | 1425 (83) |

Table 5.2: HIV-1 subtypes.

| Subtype | Training set | Test set | Total |
|---|---|---|---|
| A | 39 | 46 | 85 |
| B | 955 | 943 | 1898 |
| C | 168 | 149 | 317 |
| 02_AG | 12 | 15 | 27 |
| O | 11 | 11 | 22 |
| D | 69 | 95 | 164 |
| A1 | 25 | 18 | 43 |
| AG | 5 | 5 | 10 |
| 01_AE | 97 | 106 | 203 |
| G | 7 | 7 | 14 |
| Others | 37 | 30 | 67 |
| Total | 1425 | 1425 | 2850 |

Table 5.3: Sequence length distribution. The minimum length is 31 residues and the maximum length is 40 residues.

| Residues | Training set | Test set | Total |
|---|---|---|---|
| 31 | 1 | 0 | 1 |
| 32 | 0 | 0 | 0 |
| 33 | 2 | 2 | 4 |
| 34 | 18 | 22 | 40 |
| 35 | 210 | 189 | 399 |
| 36 | 1142 | 1162 | 2304 |
| 37 | 30 | 31 | 61 |
| 38 | 11 | 10 | 21 |
| 39 | 11 | 8 | 19 |
| 40 | 0 | 1 | 1 |
| Total | 1425 | 1425 | 2850 |

Table 5.4: Classification results on the test sets. Accuracy, specificity and sensitivity are defined in Methods. See [25] for a description of the ROC area.

| Coreceptor usage | SVM parameter C | Kernel parameter | Support vectors | Accuracy | Specificity | Sensitivity | ROC area |
|---|---|---|---|---|---|---|---|
| **B**lended spectrum kernel | | | | | | | |
| CCR5 | 0.04 | 3 | 204 | 96.63% | 85.33% | 98.75% | 98.68% |
| CXCR4 | 0.7 | 9 | 392 | 93.68% | 96.00% | 87.68% | 96.59% |
| CCR5 and CXCR4 | 2 | 15 | 430 | 94.38% | 98.16% | 67.05% | 90.16% |
| **L**ocal alignment kernel | | | | | | | |
| CCR5 | 9 | 1 | 200 | 96.42% | 87.55% | 98.08% | 98.12% |
| CXCR4 | 0.02 | 0.05 | 321 | 92.21% | 97.56% | 78.39% | 95.11% |
| CCR5 and CXCR4 | 0.5 | 0.1 | 399 | 92.28% | 97.20% | 56.64% | 87.49% |
| **D**istant segments kernel | | | | | | | |
| CCR5 | 0.4 | 30 | 533 | 96.35% | 83.55% | 98.75% | 98.95% |
| CXCR4 | 0.0001 | 30 | 577 | 94.80% | 97.56% | 87.68% | 96.25% |
| CCR5 and CXCR4 | 0.2 | 35 | 698 | 95.15% | 99.20% | 65.89% | 90.97% |
| **P**erfect deterministic classifier | | | | | | | |
| CCR5 | - | - | - | 99.15% | 99.55% | 99.08% | - |
| CXCR4 | - | - | - | 98.66% | 99.70% | 95.97% | - |
| CCR5 and CXCR4 | - | - | - | 97.96% | 99.68% | 85.54% | - |
| **D**istant segments kernel trained on test set | | | | | | | |
| CCR5 | 0.3 | 40 | 425 | 98.45% | 92.88% | 99.5% | 99.17% |
| CXCR4 | 0.0001 | 35 | 611 | 98.66% | 99.70% | 95.97% | 98.29% |
| CCR5 and CXCR4 | 0.0001 | 40 | 618 | 97.96% | 99.68% | 85.54% | 96.27% |

Table 5.5: Available methods. The results column contains the metric and what the classifier is predicting.

| Reference | Learning method | Training set | Testing set | Multiple alignments | Results |
|---|---|---|---|---|---|
| Pillai et al. 2003 | Charge rule [5, 6] | 271 | - | yes | Accuracy (CXCR4): 87.45% |
| Resch et al. 2001 | Neural networks | 181 | - | yes | Specificity (X4): 90.00% |
| Pillai et al. 2003 | SVM | 271 | - | yes | Accuracy (CXCR4): 90.86% |
| Jensen et al. 2003 | PSSM[a] | 213 | 175 | yes | Specificity (CXCR4): 96.00% |
| Jensen et al. 2006 | PSSM | 279 | - | yes | Specificity (CXCR4): 94.00%[b] |
| Sander et al. 2007 | SVM | 432 | - | yes | Accuracy (CXCR4): 91.56% |
| Xu et al. 2007 | Random forests | 651 | - | yes | Accuracy (R5): 95.10% |
| Lamers et al. 2008 | Neural networks | 149 | - | yes | Accuracy (R5X4): 75.50% |
| This manuscript | SVM | 1425 | 1425 | **no** | Accuracy (CXCR4): **94.80%** |

[a]Position-specific scoring matrices
[b]Subtype C

## 5.16 Figure legends

DISTANT-SEGMENTS-KERNEL$(s, t, \delta_m, \theta_m)$

$c \leftarrow 0$

FOR any two $j_s, j_t$ such that $s[j_s{+}1] = t[j_t{+}1]$ DO

$n \leftarrow \min(|s|{-}j_s,\ |t|{-}j_t,\ \delta_m)$

$k \leftarrow -1$ ; $i \leftarrow 1$

WHILE $i \leq n$ DO

$k \leftarrow k + 1$ ; $i_{2k} \leftarrow i$

DO $i \leftarrow i{+}1$ WHILE $(i \leq n$ AND $s[j_s{+}i] = t[j_t{+}i])$

$i_{2k+1} \leftarrow i$ ; $l_k \leftarrow i_{2k+1} - i_{2k}{+}1$

DO $i \leftarrow i{+}1$ WHILE $(i \leq n$ AND $s[j_s{+}i] \neq t[j_t{+}i])$

$c \leftarrow c + \binom{l_0}{3} - 2\binom{l_0-\theta_m}{3} + \binom{l_0-2\theta_m}{3}$

$c \leftarrow c + \min(\theta_m, i_1{-}i_0) \cdot \sum_{r=1}^{k} \left(\binom{l_r}{2} - \binom{l_r-\theta_m}{2}\right)$

RETURN $c$

Figure 5.1: The algorithm for computing $k_{DS}^{\delta_m,\theta_m}(s,t)$.

```
EXTRACT-FEATURES(s, δ_m, θ_m)
  f ← create Map
  FOR p ← 1 to |s| − 1 DO
     FOR δ ← 2 to min(|s| − p + 1, δ_m) DO
        FOR θ_α ← 1 to min(δ − 1, θ_m) DO
           FOR θ_α' ← 1 to min(δ − θ_α, θ_m) DO
              α ← s(p : p + θ_α − 1)    ; α' ← s(p + δ − θ_α' : p + δ − 1)
              key← MAKE-KEY(δ, α, α')
              IF f does not contain key THEN  f[key] ← 0
              f[key] ← f[key] + 1
RETURN f
```

Figure 5.2: The algorithm for extracting the features of a string $s$ into a Map. Here, $s(i : j)$ denotes the substring of $s$ starting at position $i$ and ending at position $j$.

EXTRACT-DISCRIMINANT($S, \delta_m, \theta_m, \alpha$)

$w \leftarrow$ create Map

FOR $i \leftarrow 1$ to $|S|$ DO

$f \leftarrow$EXTRACT-FEATURES($s_i, \delta_m, \theta_m$)

FOR EACH $key$ in $f$ DO

IF $w$ does not contain $key$ THEN $w[key] \leftarrow 0$

$w[key] \leftarrow w[key] + y_i \alpha_i f[key]$

RETURN $w$

Figure 5.3: The algorithm for merging features.

This algorithm can merge every feature from the set $S = \{(s_1, y_1), (s_2, y_2), ..., (s_l, y_l)\}$ of all support vectors into a Map representing the discriminant $w$.

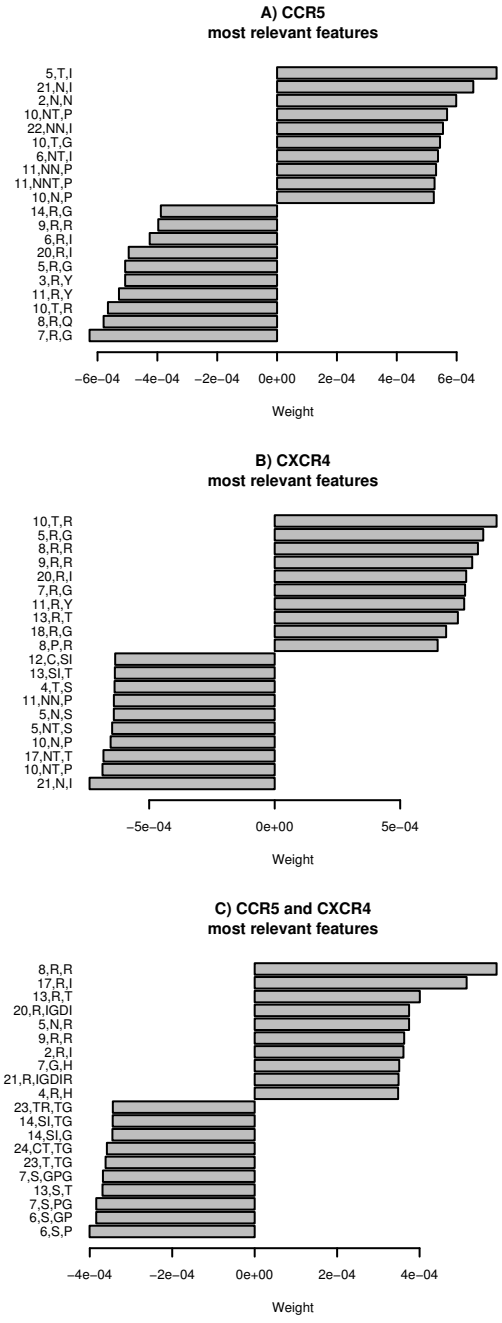


Figure 5.4: Features (20 are shown) with highest and lowest weights for each coreceptor usage prediction task.

## 5.17 Additional files

Additional file 1

File format: ZIP

Title: Source code and data.

Description: Web server, classifiers, discriminant vectors and data sets.

# Chapitre 6

# Article 2 : Ray : simultaneous assembly of reads from a mix of high-throughput sequencing technologies

## 6.1 Journal

Ce manuscrit a été publié dans le journal *Journal of Computational Biology* le 2010-10-20 pendant mon doctorat. Cet article est distribué en libre accès.

## 6.2 Résumé

Une séquence précise d'un génome d'une espèce est désormais un pré-requis pour la recherche génomique. Une étape importante dans l'obtention d'une séquence génomique de haute qualité est d'assembler correctement les courtes lectures d'ADN en séquences plus longues représentant des régions génomiques contiguës , appelées *contigs* en anglais. Les technologies de séquençage actuelles continuent d'offrir des augmentations de débit, et une réduction correspondante des coûts et de temps. Malheureusement, l'avantage obtenu avec un grand nombre de lectures est compliqué par les erreurs de séquençage, avec différents biais observés pour chaque plateforme. Bien que des logiciels sont disponibles pour assembler les lectures pour chaque système individuel, aucune procédure n'a été proposée pour l'assemblage de haute qualité basé sur un mélange de différentes technologies. Dans cet article, nous décrivons un assembleur parallèle de

courtes lectures, appelé Ray, qui a été développé pour assembler des lectures obtenues à partir d'une combinaison de plateformes de séquençage. Nous avons comparé sa performance à celle d'autres assembleurs sur des ensembles de données réelles et simulées. Nous avons utilisé une combinaison de Roche/454 et Illumina pour assembler trois génomes différents. Nous avons montré que le mélange de technologies de séquençage réduit systématiquement le nombre de séquences résultantes et le nombre d'erreurs. En raison de sa nature ouverte, ce nouvel outil, nous l'espérons, pourra servir de base pour développer un assembleur qui peut être d'une utilisation universelle.

## 6.3 Contributions

J'ai écrit le logiciel, fait les expériences bio-informatiques, et rédigé le brouillon du manuscrit. J'ai aussi soumis le manuscrit à l'éditeur et adressé les commentaires des évaluateurs.

## 6.4 Contenu

Ray: simultaneous assembly of reads from a mix of high-throughput sequencing technologies

Sébastien Boisvert[1,2,*], François Laviolette[3], Jacques Corbeil[1,2]

[1] Département de médecine moléculaire, Université Laval, 1050, avenue de la Médecine, Québec (Québec), Canada, G1V 0A6

[2] Infectiologie et immunologie, Centre de recherche du CHUQ, 2705, boulevard Laurier, Québec (Québec), Canada, G1V 4G2

[3] Département d'informatique et de génie logiciel, Université Laval, 1065, avenue de la Médecine, Québec (Québec), Canada, G1V 0A6

[*] Corresponding author

## 6.5 Abstract

An accurate genome sequence of a desired species is now a pre-requisite for genome research. An important step in obtaining a high-quality genome sequence is to correctly assemble short reads into longer sequences accurately representing contiguous genomic regions. Current sequencing technologies continue to offer increases in throughput, and corresponding reductions in cost and time. Unfortunately, the benefit of obtaining a large number of reads is complicated by sequencing errors, with different biases being observed with each platform. Although software are available to assemble reads for each individual system, no procedure has been proposed for high-quality simultaneous assembly based on reads from a mix of different technologies. In this paper, we describe a parallel short-read assembler, called Ray, which has been developed to assemble reads obtained from a combination of sequencing platforms. We compared its performance to other assemblers on simulated and real datasets. We used a combination of Roche/454 and Illumina reads to assemble three different genomes. We showed that mixing sequencing technologies systematically reduces the number of contigs and the number of errors. Because of its open nature, this new tool will hopefully serve as a basis to develop an assembler that can be of universal utilization. Availability: http://deNovoAssembler.sf.Net/.

Keywords: Genome assembly, high-throughput sequencing, de Bruijn graphs

## 6.6 Introduction

The availability of high-throughput sequencing has altered the landscape of genome research. The sequencing technologies provide a tremendous amount of sequences at a lower cost with increasing lengths of reads [184]. With these sequences gathered, computational methods enable genome-wide analysis, comparative genomics studies, the hunt for resistance mutations, and the general study of molecular evolution amongst others with the constant of requiring accurately-assembled genomes. To decode a genome, its DNA is sheared in fragments, which are decoded with sequencers to produce reads. Reads are assembled by assemblers into larger sequences called contigs.

Three types of assemblers were previously introduced – greedy assemblers, overlap-layout-consensus assemblers and de Bruijn assemblers [229]. Owing to a minimal overlap length between reads, overlap-layout-consensus assemblers are unable to assemble short reads.. The dated assemblers tailored for Sanger sequencing were not designed to tackle the large datasets produced by the new sequencing technologies [229]. Although the de Bruijn framework was described as a reliable workhorse for Sanger sequencing projects [218], few de Bruijn assemblers were available when high-throughput sequencing became widely available. The de Bruijn graph was deemed well-suited for assembly of short reads [46]. A de Bruijn graph allows a compact representation of millions of short ($< 50$ nucleotides) reads [307], where short sequences ($k$-mers) occuring in reads are only stored once.

Numerous assemblers for high-throughput sequencing are readily available, but none can simultaneously assemble reads from different sequencing technologies. On the other hand, Ray can simultaneously assemble in parallel reads from a mix of sequencing systems.

## 6.7 The assembly problem

In a typical setting, genome fragments are sequenced to produce reads, followed by de novo assembly of these reads. A contig is a contiguous sequence produced by an assembler, and represents a region of the genome. A contig can contain errors – their types are described below. The length of a contig is counted in nucleotides (nt) or basepairs (bp) when we infer the double-stranded DNA molecule. Each algorithm that solves the assembly problem (AP) constructs contigs from a family of reads. We use the terminology of family instead of set because a read can appear more than once in a family.

For a one-chromosome genome, one strives to obtain a contig that is exactly the genome sequence, but this is nearly impossible in practice because of long repeated regions. A given contig can be compared to the genome sequence to find assembly errors. Among contigs that deviate from the genome sequence, one can find the following error types. A chimeric contig (or incorrect contig) contains two (or more) sequences that are not contiguous in the genome. A mismatch occurs in a contig at a particular position if it aligns on the genome sequence, but at this position the letter is not the same. An insertion in a contig is characterized by the addition of a (short) sequence of letters at some position, in respect to the genome sequence. In a similar fashion, a deletion in a contig is characterized by the removal of a (short) sequence of letters at some position, in comparison with the genome sequence. An indel is an insertion or a deletion.

A read can be sampled from the forward strand or the reverse-complement strand in the genome. Reads are generated from the alphabet $\{A, T, C, G, N\}$, where N represents ambiguous bases. A read is, in general, short (36 nt to 420 nt) compared to even the smallest bacterial genomes (2,000,000 nt), and individual reads may contain sequencing errors.

In addition, several protocols exist to sequence the extremities of longer molecules in order to generate paired reads. They are critical to genome assemblies, as they provide linkage between sequences separated by a variety of physical distances. For a fragment library, we denote the average length of such molecules by $d_\mu$ and the standard deviation with $d_\sigma$. For each library, $d_\mu$ and $d_\sigma$ are calculated by Ray automatically. Paired reads provide knowledge of the sequence content of each of the reads in the pair, but also knowledge of the orientation (DNA strand) of each read, and an estimation of the distance (amount of intervening sequence) between them [7].

The quality of a specific assembly must be evaluated through some criteria. The length of the contigs is of course one of them. This leads us to the following definition of the assembly problem.

**Definition 1.** *Given a family of reads (which can contain errors, be paired, and be from different sequencing technologies), the assembly problem (denoted AP) consists in constructing contigs such that:*

1. *the breadth of coverage (how much of the genome is represented) is maximal;*
2. *the number of assembly errors (chimeric contigs, mismatches, insertions, and deletions) is minimal;*
3. *the number of contigs is minimal.*

The minimality of the number of contigs indirectly minimizes the redundancy, i.e. contigs whose positions in the genome sequence overlap. A contig can be interpreted as a mosaic of reads and the depth of coverage of a position is the number of reads covering it in the mosaic. Uneven genome coverage can complexify the assembly problem, because the classification of a region as being repeated or not is more difficult. In the assembly problem, it is assumed that, on average, each position is covered numerous times ($> 10$X). This allows us to expect that most of the positions will be covered in multiple independant reads, and helps resolve sporadic sequencing errors in occasional reads. The difficulty of the assembly problem lies on the quality of the reads, the uneven coverage, and the complexity of the genome (large in size, non-random sequence composition, and repetitive elements often longer than existing read lengths).

## 6.8 Sequencing technologies

Genome variation is responsible for biological diversity and sequencing enables its assessment. Sequencing started with the chain-terminating inhibitor method by [262] in which subsequences are randomly stemmed from a sequence template by targeted polymerase chain reaction, which is coupled with chain termination. This method was automated with a computer and fluorescence detection [279]. The newly-developed concepts of massively-parallel flow-cell sequencing and sequencing-by-synthesis were fundamental shifts in genomics (see [117] for a review). The new sequencing technologies – Roche/454, Illumina, and SOLiD – produce shorter reads in comparison with the Sanger method. Other forthcoming sequencing technologies promise longer reads. Short reads are sufficient for de novo sequencing if they are paired[9] . Emerging sequencing technologies are evolving fast – the continuing evolution of these technologies, and others in development, is a strong argument for the need of assemblers that are technology (and read length) independant.

### 6.8.1 Sequencing errors

Each sequencing technology makes specific errors. The mismatches in Roche/454 and Illumina reads occur mostly at random and the most common Roche/454 sequencing errors are carry-forward errors (because of leftover unincorporated nucleotides), and homopolymer-related errors (because of the absence of reversible terminators) [250]. Some of these errors are correlated (the errors in homopolymer runs are correlated), which makes the AP harder, and leads us to consider algorithms that run on mixed technologies. In the Illumina technology, one nucleotide is incorporated in each cycle (because of reversible terminators), regardless if homopolymers are present. In contrast, the Roche/454 system allows the incorporation of all nucleotides of a homopolymer during the same cycle. The homopolymer problem arises because the intensity of emitted light is linear when the number of incorporated nucleotides in the homopolymeric region is low, but is not linear when too many nucleotides are incorporated at once.

## 6.9 Assembly algorithms

Since the release of the first assembled genome, computer-aided methods have evolved along with the evolution of sequencing apparatus. Over the course of this evolution, a number of algorithmic pathways were followed and quickly rendered obsolete by the sheer volume outputted by the new parallel sequencing technologies. Available assemblers are dedicated to specific and proprietary technologies. Our assembler Ray is, on the contrary, not dedicated to any specific technology, parallel, and open source. One aim of our algorithm is to allow the simultaneous use of reads produced by different technologies. What follows is a brief description of assembler types, along with a list of the most important related algorithms.

### 6.9.1 Overlap-layout-consensus

Intuitively, finding read overlaps should be the primary idea behind assembly. The overlap-layout-consensus framework is merely a three-step process [195]. First, the overlaps between reads are computed. Second, the order and orientation of reads in the assembly are determined. Third, the consensus allows the determination of the nucleotide at each position in the contigs.

Assemblers implementing this idea are numerous, and were crafted to overcome the assembly hurdles in Sanger-technology projects before high-throughput systems were developed. These software are tailored for the assembly of long reads, such as Sanger reads. They include the Celera assembler [195] and Arachne [21]. Afterwards, the paradigm was adapted to the Roche/454 system. The Roche/454 sequencer is distributed with Newbler [24].

Also, EDENA is an overlap-layout-consensus assembler that can assemble short reads (35 bases) [110].

### 6.9.2 Greedy assemblers

Using overlaps between reads to build contigs is intuitive. Algorithms can be built upon this idea. A greedy algorithm iteratively grows contigs by choosing best overlaps first. Implementations of the greedy algorithm were the first to be introduced for the short read technologies: SSAKE [296] VCAKE [130] , and SHARGCS [72].

### 6.9.3 Assembly with de Bruijn graphs

The introduction of the de Bruijn graph for the AP is motivated by redundant information in reads: a large depth of coverage implies that a lot of overlaps between reads occurs. In presence of such redundant information, de Bruijn assemblers can solve the assembly problem with a memory usage bounded by the genome length [307].

Before introducing the de Bruijn graph, we need to give some definitions. For the alphabet $\Sigma = \{A, T, C, G, N\}$, $\Sigma^*$ is the associated set of strings. For a string $x$, $|x|$ denotes its length, $x[i]$ is the letter (or symbol) at position $i$ (starting at 1), and $x[i..j]$ is the substring of $x$ from positions $i$ to $j$. For strings $x$ and $y$, $xy$ denotes their concatenation.

We denote the reverse-complement sequence of $x \in \Sigma^*$ with revcomp$(x)$. The definition is as follows: for a sequence $x$, its reverse-complement, $y = \text{revcomp}(x)$, is defined as $\text{complement}(y[i]) = x[|x| - (i-1)]$, where the complement is defined as usual: $\text{complement}(A) = T$, $\text{complement}(T) = A$, $\text{complement}(C) = G$, $\text{complement}(G) = C$ and $\text{complement}(N) = N$. Here is an example: $\text{revcomp}(AGGGAT) = ATCCCT$.

Finally, a $k$-mer is a sequence of length $k$ generated with the alphabet of nucleotides. Using these concepts and notations, we define the de Bruijn graph as follows.

**Definition 2.** *Given an alphabet $\Sigma = \{A, T, C, G\}$ and a integer $k$, a full de Bruijn graph $G$ has vertices $V(G) = \Sigma^k$, and arcs $\{\langle x, y\rangle,\ x, y \in V(G),\ x[2..k] = y[1..k-1]\}$*

where $\Sigma^k$ is the set of all possible $k$-mers with $\Sigma$. Note that, in a de Bruijn graph, a vertex represents a sequence of $k$ letters (a $k$-mer), an arc, being composed of two $k$-mers that overlap on $k-1$ letters, can be viewed as a sequence of $k+1$ letters (a $(k+1)$-mer), and more generally, a walk of $n$ vertices corresponds to a sequence of $n+k-1$ letters. A walk, in an oriented graph, can contain several times the same vertex, akin to walking on the graph respecting the orientation of the arrows. In the de Bruijn framework, a genome sequence of $l$ letters is a walk in the de Bruijn graph, and has $l-k+1$ vertices. We have a similar observation for contigs. In the case of reads, the same holds with the exception that $k$-mers containing the symbol N are not in the corresponding walk.

Given a full de Bruijn graph $G$, and a family of reads $D \subseteq \{A, T, C, G, N\}^*$, one can build a subgraph of $G$ by retaining only the vertices and the arcs in the graph that correspond respectively to $k$-mers and $(k+1)$-mers present in the reads of $D$, or in

reverse-complement reads of $D$. The value of the integer $k$ is a parameter that has to be fixed for each de Bruijn algorithm, and 19 is the lowest sensible value (we use 21) [13]. With this value, we have a rich set of vertices in the full de Bruijn graph ($4^{21}$ = 4,398,046,511,104). Moreover, $k = 21$ is significantly shorter than the length of the reads given by the present sequencing technologies. Recall that for any read, the strand on the genome sequence from which it was generated is unknown. de Bruijn graphs based on reads of a family $D$ will therefore be constructed based on all the reads of $D$ together with all the reverse-complement reads of $D$.

In the particular case of a pair of reads ($r_1$,$r_2$,$d_\mu$,$d_\sigma$), four subsequences will be considered for the constructed de Bruijn graph, namely $r_1, r_2$ and their reverse-complement sequences, where $r_1$ and $r_2$ are read sequences of a fragments library, and $d_\mu$ is the average fragment length observed for the library (with standard deviation $d_\sigma$).

If the reads contain errors, particular structures in the graph – bubbles and tips – will be present [307]. A bubble is a structure between two vertices, such that two disjoint (short) walks exist between these vertices, and the sequence of letters of each walk is very similar. A tip is a hanging walk in the graph leading to a dead-end, and whose length is short.

In the case where reads are error-free and cover the whole genome and when there are no repeated sequences with more than $k$ letters, the solution of the AP is a single walk in the de Bruijn graph that goes through each arc exactly once. Such a walk is called Eulerian. In the case where the two strands of the genome have been sequenced without errors, the solution is not a single walk, but the union of two walks which collectively use each edge exactly once.

The use of Eulerian walks was previously described as a solution for the assembly problem [124, 92, 125, 218]. Eulerian walks can be found in polynomial time, but their application to high-throughput sequencing was not particularly successful for making practical assemblies [13]. Basically, the EULER approach makes a series of equivalent transformations that lead to a final set of paths [218]. With this idea, even if a $(k+1)$-mer is present at two or more locations in the genome sequence, it is still possible to construct a single Eulerian walk, using edge multiplicities. However, to determine for each edge which multiplicity is suitable represents a difficult problem when the reads give an uneven coverage.

Many strategies (other than Eulerian walks) have been proposed to produce a good

assembly with a de Bruijn graph . They are based on the idea of simplifying the de Bruijn graph . This simplification addresses the presence of topological errors in the graph and repetitive sequences.

We describe the existing algorithms in Section 6.9.3 and their respective problems in Section 6.9.3. Note that, in contrast, our proposed algorithm neither limits its search to Eulerian walks, nor performs such simplifications on the de Bruijn graph. Indeed, as shown in Section 7.8, we not only keep the de Bruijn graph as it is, but we even add annotations to preserve, as far as possible, all read information. Then, our approach searches for walks that are compatible with the given family of reads.

### de Bruijn algorithms

The pioneering work in this field [218] proposed an algorithm that searches for Eulerian walks in a de Bruijn graph. Afterwards, this approach was adapted to short reads [46]. With the advance of high-throughput sequencing, Chaisson et al. created the EULER-SR framework which deals with short reads and high-coverage data [49, 9]. Another de Bruijn assembler, called Velvet [307], was successful at assembling short reads. Velvet builds a de Bruijn graph, and transforms it in a simplified de Bruijn graph. In Velvet, the graph is also corrected by removing tips and by extracting sequence information from bubbles.

Furthermore, ABySS assembled the human genome with a parallel approach [276]. ALLPATHS builds a graph and finds all walks between each paired reads [39]. Another approach for the assembly of very short reads (25 nt) was described by [118]. In their work, the authors developed SHORTY, which is motivated by the concept of seeds. The latter are long (500 nt) high-quality sequences from which the assembly can stem. In our proposed algorithm, we also make use of the concept of seeds, but these seeds are computed by Ray with solely the short reads provided (in SHORTY, seeds are provided by the user). Finally, note that each of these algorithms only runs on a single technology. And only ABySS and Ray run in parallel and distribute data across computers with message passing interface.

### Problems with current de Bruijn assemblers

To address the errors in the graph, a coverage cutoff can be defined to trim erroneous artifacts that are interspersed in the de Bruijn graph [307]. Assembly protocols set the coverage cutoff to a particular value and erode anything in the graph whose coverage is

lower than the cutoff. It is unlikely that all the vertices that have a depth of coverage lower than the cutoff correspond to concealed errors. Thus, such a cutoff will prohibit an assembler from yielding large contigs.

Existing assemblers are not designed for the assembly of mixes of reads from different systems, being only able to run on a single technology. Also, only ABySS and Ray calculate assemblies in parallel using message passing interface.

## 6.10 Mixing sequencing technologies

Using a mix of sequencing technologies is called hybrid assembly, and decreases the number of correlated errors in the reads [69]. In principle, mixing sequencing technologies improves de novo assemblies. No solution has been designed to specifically and efficiently tackle this problem. Current hybrid-assembly procedures are not assembling simultaneously reads acquired from different systems [98, 13, 69]. The asynchronous fabrication of contigs does not highlight the errors contributed by each single sequencing system. Moreover, not piecing together reads simultaneously can lead to loss of valuable information about short repeated regions [49]. To address these shortcomings, Ray makes concurrent use of all available reads. In our experiments, namely in SpErSim (see Section 6.12), we observed that the assembly problem is simpler when the sequencing errors are based on pure random noise. Hence, we consider that mixing technologies is a good compromise to mimic as closely as possible randomly-occuring noise. The emergence of an array of sequencing technologies makes this compromise realistic.

### 6.10.1 Hybrid assembly methods

[98] have assembled Sanger and Roche/454 reads together to produce high-quality draft assemblies. However, the lack of tools to perform simultaneous hybrid assembly was translated in the usage of more than one assembler. They assembled the Roche/454 reads with a first assembler, created Sanger-like reads from the Roche/454 contigs, and assembled the latter with Sanger reads with a second assembler.

While Roche/454 reads are sufficient to produce high-quality drafts, the latter contain small insertions/deletions, and mismatches that are directly caused by the technology used, and therefore are correlated among reads. To improve the rough quality of a draft, sequences from another technology can be aligned onto the draft assembly to

find errors [13]. This method is not a truly hybrid assembly because the Roche/454 and Illumina reads are not simultaneously assembled. Note also that in their paper, the authors suggested that beyond a particular coverage with short Illumina reads, the ability to correct errors present in a Roche/454 assembly can not be significantly improved, which is in contradiction with our experimental results. Indeed, with Ray, we experimentally demonstrated that it is possible to avoid almost all assembly errors by simultaneously assembling the Roche/454 and Illumina reads (see Section 6.12).

[69] have gathered capillary (Sanger), Roche/454, and Illumina reads to decode the genome of a 32.5 Mb filamentous fungus. Although mixing sequencing technologies is strongly promoted by these authors, the underlying assembler, Forge, is unable to work directly with Illumina reads. The latter were preprocessed, using the Velvet program, to obtain sequences usable by Forge. This dataset is not considered herein because of the lack of a finished sequence.

## 6.11 The algorithm Ray

Ray is derived from the work by [218] in the sense that we make use of a de Bruijn graph. However, our framework does not rely on Eulerian walks. To generate an assembly, we define specific subsequences, called seeds, and for each of them, the algorithm extends it into a contig. We define heuristics that control the extension process in such a way that the process is stopped if, at some point, the family of reads does not clearly indicate the direction of the extension. In example, consider the graph of Figure 6.1 and suppose that the seed is $z_2 \cdots z_3$, then if most of the reads that overlap the seed also contain $z_5$, then we will proceed by adding $z_5$ to the contig. On the contrary, we will choose an another direction, and in the case where there is no obvious winner, the process will stop. Note that our heuristics will give a higher importance to reads that heavily overlap a contig than to reads that only intersect a small number of the last vertices of it. Indeed, we consider that reads that overlap strongly in the contig we are constructing are more informative to assign the direction in which the extension should proceed.

We measure the overlapping level with functions called $\text{offset}_i$ and $\text{offset}_i^{\text{paired}}$. The proposed heuristics will therefore limit the length of the obtained contigs, but will give better guarantees against assembly errors. Moreover, those heuristics allow us to construct greedy algorithms in the sense that at each point, we extend the contig that we are constructing, or we end the construction. Such a greedy choice that will never

be reconsidered is inevitable if one wants to have an algorithm that runs in polynomial time. The choice in favor of greedy optimization is motivated by the NP-hard nature of the problem [229].

To minimize running time and memory utilization, we use the sequence data provided by the reads through a de Bruijn graph annotation, each read being annotated only once and attached to a single vertex of the graph.

### 6.11.1 Coverage distribution

Given a family of reads $D = \langle r_1, \ldots, r_t \rangle$, we denote by $D^+$ the family consisting of the reads of $D$ and the reverse-complement sequences of $D$, or more precisely, $D^+ = \langle r_1, \ldots, r_t, \text{revcomp}(r_1), \ldots, \text{revcomp}(r_t) \rangle$. Moreover, for any integer $c$, a $k$-mer is $c$-confident if it occurs at least $c$ times in the reads of $D^+$. Note that a $k$-mer that appears twice in the same read is counted twice. We define $f_{D^+}$ as the function that associates to each integer value $c$ the number of $k$-mers that are covered $c$ times according to $D^+$. As proposed by [49], this function is the sum of an exponentially-decreasing curve and a Poisson distribution. The first represents the errors contained in the reads, and the second is a Poisson distribution because each read is a subsequence picked among all subsequences around a particular length with some almost uniform probability. The graph of the function $f_{D^+}$ has the shape of the function drawn in Figure 6.2 – the coverage distributions for the *A. baylyi* ADP1 dataset are shown in this figure, with Roche/454 reads, Illumina reads, and a mix of those. $f_{D^+}$ has a local minimum, that we will call $c_{min}$, followed by a local maximum, that we will denote $c_{peak}$. Basically, $c_{peak}$ is an estimate of the average coverage depth of the genome, and $c_{min}$ is an estimation of the value where, with high probability, the amount of incorrect data is lower than the amount of correct data.

### 6.11.2 Annotated de Bruijn graph

Given a family of reads $D$, a de Bruijn parameter $k$, and a coverage cutoff $c$, the associated annotated de Bruijn graph, noted $Bruijn(D^+, k, c)$ is defined as follows. The vertices $V$ of the graph are the $c$-confident $k$-mers and the set of arcs $A$ is composed of all arcs between two vertices of $V$ that correspond to a 1-confident $(k+1)$-mer. For example, for $k = 5$ and $c = 2$, the two $k$-mers $ATGCT$ and $TGCTG$ will be vertices of the graph if and only if they both appear at least twice in the reads of $D^+$. Moreover, $\langle ATGCT, TGCTG \rangle$ will be an arc of the graph if and only if $ATGCTG$ appears at

least once in the reads of $D^+$.

For the annotation, we consider the first $c$-confident $k$-mer of each read $r$; we will denote this $k$-mer $x_r$. We then remove from $D^+$ every read $r$ for which the associated $x_r$ has a confidence value of 255 or more. A vertex with a confidence value of 255 or more is repeated because any region of a genome is usually covered 30-50 times. We consider that any $x_r$ for which we have such coverage will belong to a repeated region. These regions are more susceptible to generate misassemblies. We avoid these regions, and do not consider the corresponding reads for the annotation. This choiceimplies that the algorithm may output contigs of shorter length, but we will gain on the quality of these contigs.

For the remaining $x_r$'s, first note that $x_r$ is a subsequence of the read $r$ that is of length $k$ and such that any subsequence of length $k$ that starts before $x_r$ in $r$ is not $c$-confident. To each such remaining $x_r$, we complete the annotation $(r, s)$, where $s$ is the starting position of subsequence $x_r$ in the read $r$. Hence, on a given vertex $x$, one can recover any read for which $x$ is the first $c$-confident $k$-mer. Note that this annotation system is not memory intensive since each read is annotated only once. Moreover, the other positions of the read are recovered by the algorithm using the following functions.

**Offset functions**

Given a walk $w = \langle x_1 \cdots x_l \rangle$, and an arc $\langle x_l, y \rangle$, we define the i$^{\text{th}}$ offset value of $y$ according to $w$, noted $\text{offset}_i(w, y)$, as the number of annotations $(r, s)$ on the vertex $x_i$, such that $y$ is a subsequence of $r$, and such that the distance between the starting point of $x_i$ and the starting point of $y$ are equal in both the read $r$ and the walk $w$ concatenated with the vertex $y$. The starting position of $y$ in the read $r$ is $l - i + s + 1$.

As an example, put $k = 5$ and $c = 1$, and consider that we have a single read $r = ATGCATCG$ in $D$. Since $c = 1$, all the 5-mers of $r$ belong to the de Bruijn graph. Also, on the vertex $x_r = ATGCA$, we will have the annotation $(r, 1)$. Now consider the walk $w = \langle ATGCA, TGCAT, GCATC \rangle$, and the vertex $y = CATCG$. Then, we have that $\text{offset}_1(w, y) = 1$ because the read $r$ starts at position 1 in $w$, and the distance between $y$ and $x_r$ is the same in the read and in the concatenation of the walk $w$ with the new arc $\langle GCATC, CATCG \rangle$. Note that $y$ is a subsequence of $r$, whose starting position is 4, which is exactly $l - i + s + 1$, since the length of the walk $w$ is $l = 3$, the

position $x_r$ in $w$ is $i = 1$, and the position of $x_r$ in the read $r$ is $s = 1$.

We will have the same type of functions for paired reads, except that since the fragment length is approximated by $d_\mu$ for each library, we will tolerate a certain difference between the length $d$ of a particular pair of reads and the average length $d_\mu$ for the corresponding fragment library – namely $d_\sigma$. More formally, for a read pair $(r_1, r_2, d_\mu, d_\sigma)$, let us define the meta read $r_{12}$ which has length $d$, begins with $r_1$, ends with revcomp($r_2$), and is composed of dummy letters in the middle. Then, given a walk $w = \langle x_1 \cdots x_l \rangle$, and an arc $\langle x_l, y \rangle$, we define the i$^{\text{th}}$ paired offset value of $y$ according to $w$, noted $\text{offset}_i^{\text{paired}}(w, y)$, as the number of annotations $(r_1, s_1)$ on the vertex $x_i$, for which there exists a paired read $(r_1, r_2, d_\mu, d_\sigma)$, such that $y$ is a subsequence of revcomp($r_2$), and the distances between the starting point of $x_i$ and the starting point of $y$ in the meta read $r_{12}$ and in the walk $w$ concatenated with the vertex $y$ differ by a value that is at most $d_\sigma$. Mathematically, the starting position of $y$ in the read $r_2$ is . $d_\mu - |r_2| - (l - i + s_1 + 1) \pm d_\sigma$.

As stated above, we want to give more importance to reads that have a large overlap with the contig we are constructing. Let us point out that when the value of $i$ is getting smaller in the offset function, the magnitude of the overlap is increasing. So given a partially constructed contig $w = \langle x_1, \cdots, x_l \rangle$, and two possible arcs $\langle x_l, y \rangle$ and $\langle x_l, y' \rangle$ that we can add to it, we will compare the offset values of $y$ and $y'$ according to $w$, if there is a clear winner, the algorithm will choose it, otherwise it will stop the ongoing extension.

### 6.11.3 Seeds

The proposed algorithm starts on some walks, for which we have a strong confidence that they are subsequences of the genome sequence. Such particular walks are referred to as seeds. To obtain them, we first consider a de Bruijn graph with a very high cutoff value. We fix this cutoff to $\frac{c_{min}+c_{peak}}{2}$. $c_{peak}$ corresponds to the average coverage and $c_{min}$ to where the number of errors is lower than the number of true sequences. We consider that by taking the average of those two, a very small amount of errors will be in the de Bruijn graph $Bruijn(D^+, k, \frac{c_{min}+c_{peak}}{2})$. The resulting stringent de Bruijn graph suffers from a huge loss of the information contained in the reads, but this procedure is only used to determine the seeds. More specifically, the seeds are the set of maximum walks in $Bruijn(D^+, k, \frac{c_{min}+c_{peak}}{2})$ that are composed of vertices of indegree and outdegree at most one.

### 6.11.4 The heuristics

In this section, we present the heuristics on which our algorithm is based. In these rules, $m$ is a multiplicative value indicating the confidence required to perform a choice – as the coverage increases, its value decreases. For a vertex, $m = 3.0$ when its coverage is at least 2, but no more than 19, $m = 2.0$ when its coverage is at least 20, but no more than 24, and $m = 1.3$ when its coverage is at least 25. For any vertex, the relation between its coverage and the value of the multiplicator $m$ is not a parameter, and works well on various tested datasets (see Tables 3, 4, and S2-S5). The values taken by $m$ are motivated by the distribution of reads on a seeds – two reads starting consecutively on a seed will be nearer if more local coverage is available.

Given a walk $w = \langle x_1, \cdots, x_l \rangle$, and two arcs in the de Bruijn graph $Bruijn(D^+, k, c)$ that start at $x_l$, namely $\langle x_l, y \rangle$ and $\langle x_l, y' \rangle$, we say that

**Rule 1 :** $y$ wins over $y'$ if the three following inequalities hold.

$$\begin{aligned} m \cdot \sum_{i=1}^{l} (l-i) \cdot \text{offset}_i^{\text{paired}}(w, y') &< \sum_{i=1}^{l} (l-i) \cdot \text{offset}_i^{\text{paired}}(w, y) \\ m \cdot \sum_{i=1}^{l} \text{offset}_i^{\text{paired}}(w, y') &< \sum_{i=1}^{l} \text{offset}_i^{\text{paired}}(w, y) \\ m \cdot \min_{i \in \{1..l\}} \Big( \text{offset}_i^{\text{paired}}(w, y') > 0 \Big) &< \min_{i \in \{1..l\}} \Big( \text{offset}_i^{\text{paired}}(w, y) > 0 \Big) \end{aligned}$$

So, according to Rule 1, $y$ wins over $y'$ if 1) the number of paired reads that strongly overlap $y$ and $w$ is more than $m$ times the corresponding value for $y'$, 2) the total amount of paired reads that overlap $y$ and $w$ is more than $m$ times than the same amount for $y'$, and 3) the paired read that has the maximal overlap over $w$ and $y$ is more than $m$ times the same value for $y'$. Recall that the smallest is the value of $i$, the biggest is the overlap for the corresponding reads.

**Rule 2 :** By replacing the functions $\text{offset}_i^{\text{paired}}$ by their corresponding functions $\text{offset}_i$ in the three equations above, we get Rule 2.

Hence, Rule 2 is doing exactly the same type of measurement as Rule 1, except that it works on single reads.

These rules are utilized to extend seeds in the Ray algorithm. The Ray algorithm is presented in Figure 6.3.

### 6.11.5 Technical points about the implementation

The data structure selected to extract $k$-mer information in reads is the splay tree [35] – a balanced binary search tree based on the splay operation. $k$-mers are stored on 64 bits – thus, the maximum value for $k$ is 32.

## 6.12 Results and discussion

Ray compares favorably with current algorithms. We emphasize that only Ray performs very well on mixed datasets, and that the parallel nature of Ray makes it scalable. The goal of this evaluation is to demonstrate that Ray helps to assemble genomes using high-throughput sequencing. Comparisons with available assemblers are in the Supplementary Material available on the publisher website (Tables S1-S5).

We added additional mixed datasets as requested by Referee 1.

### 6.12.1 Datasets

Datasets are presented in Table 6.1. For our datasets, we selected living organisms for which high-quality data were available – these included *Streptococcus pneumoniae*, *Escherichia coli*, *Acinetobacter baylyi*, and *Cryptobacterium curtum*. *Streptococcus pneumoniae* is a significant cause of human diseases, and the avirulent strain R6 (NCBI accession: nuccore/NC_003098) is a platform for investigation [18]. *Escherichia coli* K-12 (NCBI accession: nuccore/NC_000913) is a model organism [6]. We gathered Roche/454 reads [189] and Illumina reads (produced by Illumina inc.) for *E. coli* K-12 strain MG1655. *Acinetobacter baylyi* ADP1 (NCBI accession: nuccore/NC_005966) is highly competent for natural transformation, making it very convenient for genetic engineering [3]. We downloaded Roche/454 and Illumina reads for *A. baylyi* ADP1 to test Ray [13]. *Cryptobacterium curtum* (NCBI accession: nuccore/NC_013170) is an opportunistic pathogen, and its genome was assembled with Roche/454 reads and Sanger reads [25].

We simulated two error-free datasets from *Streptococcus pneumoniae* R6 (SpSim and SpPairedSim). We also simulated one error-prone dataset from *Streptococcus pneumoniae* R6 (SpErSim). As pointed out by [118], it is striking how the presence of errors in real-world sequencing data is underestimated by the common use of simulated data. To test this, we tested Ray on real datasets. We downloaded datasets from the Short Read Archive (SRA – http://www.ncbi.nlm.nih.gov/sra). These sequence reads are

from *Escherichia coli* K-12 MG1655 [189], *Acinetobacter baylyi* ADP1 [13] , and Cryptobacterium curtum DSM 15641 [25]. For each of these genomes, the corresponding dataset contained Illumina reads, and Roche/454 reads. Table S1 describes datasets on which comparisons presented in Table S2-S5 were performed.

### 6.12.2 Quality assessment metrics

To assess the quality of assemblies and compare our method with other approaches, we selected a set of metrics usually utilized when comparing sequence quality. These metrics are described in the definition of the AP (see Definition 1), and characterize a fine assembly. They are the number of contigs having at least 500 letters (or bp), the number of bases (bp), the mean size of contigs (bp), the N50 (the length of the largest contig such that all contigs with at least its length contains at least 50% of the assembly, in bp), the largest contig size (bp), the genome coverage (breadth of coverage – percentage of bases in the genome covered), incorrect contigs, mismatches, and indels. The number of contigs is a measure of the fragmentation of the assembly. A higher number of assembly bases indicates that the assembly contains two strands for particular genomic regions, or that the DNA molecules were contaminated. The distribution of contig lengths is evaluated with the mean contig length, N50, and largest contig length. The genome coverage is crucial in evaluating the fraction of the genome being covered by the assembly. The number of contigs, mean size, N50, largest contig length, and the breadth of coverage are standard indicators, but they miss the assembly errors. The sensitivity of an assembly software towards repetitive elements – the main source of misassemblies – is evaluated with the number of incorrect contigs. Mismatches, small insertions, and small deletions are local errors. Lowering the number of these error types is paramount to the analysis of genome variation. The number of incorrect contigs, the number of mismatches, and the number of indels allow critical assessment of genome assemblies. Assembly contigs were aligned to a reference, and these alignments were evaluated using the metrics described above. Alignments of sequences were done with Mummer [143] and Blast [8].

### 6.12.3 Genome assemblers

Numerous assemblers are available. We selected the most recent assemblers in our benchmarks. We compared Ray with Velvet [307], EULER-SR [9], ABySS [276], and Newbler [24]. We refer the reader to Table 6.2 for assembler descriptions. ABySS was run according to the documentation provided. EULER-SR was run with $k = 21$ . A

rule file was provided when paired information was available. Newbler was run with default parameters using the command-line interface. Ray was run with the sequence files as input, and we fixed the parameters to the same value on all datasets, namely we fixed $c = 2$ and $k = 21$. The values of $c_{min}$ and $c_{peak}$ were calculated automatically (see Section 6.11.1), and so were the average fragment length and standard deviation for paired libraries. Ray is an assembler implementation using auto-calculated parameters ($c_{min}$, $c_{peak}$, and $d_{\mu}$ and $d_{\sigma}$ for each paired library) which renders the process readily applicable. Velvet was run with hash size 21, and the expected and cutoff coverages were determined according to the manual.

### 6.12.4 Simulations

Results on simulated data are shown in Tables 6.3 and S2. We chose to include simulated data because in the case of real data, the reference is not always error-free. Three datasets are presented. SpSim is an error-free dataset with 50-bp reads. SpErSim contains the same reads as SpSim, in addition of random mismatches. SpPairedSim is similar to SpSim, but includes paired information.

**SpSim (simulated)** The SpSim dataset contained simulated short 50-base reads at 50 X (depth of coverage) from the 2038615-base *Streptococcus pneumoniae* R6 genome. The peak coverage $c_{peak}$ in the coverage distribution had a value of 29, and the $c_{min}$ was not relevant in this dataset (thanks to the absence of errors). All five assemblers, except Newbler, were able to analyze short reads (Tables 3 and S2). In Newbler, overlaps were detected using a minimum overlap length, and this minimum overlap length was longer than the read length. The assemblies were similar, but emoveOpenAssembler Ray was slightly better for this dataset. Indeed, emoveOpenAssembler Ray produced 259 contigs, and its assembly covered 96% of the genome. Both ABySS and emoveOpenAssembler Ray produced no incorrect contigs. EULER-SR produced more mismatches than the other assemblers and yielded indels on this error-free data set. emoveOpenAssembler Ray had the best overall performance on this dataset.

**SpErSim (simulated, 1% random mismatch)** The results for this dataset were similar to those obtained with the dataset above. From these results, we concluded that assemblers are not very sensitive to random noise. The addition of errors, however, increases the amount of consumed memory, owing to the additional $k$-mers occuring in the reads, but not in the genome. These errors are instanciated as bubbles and tips in the graph.

**SpPairedSim (simulated)** This dataset included short 50-base reads at 50 X from *S. pneumoniae* R6, and these reads were generated in a paired-end fashion. The paired reads are the extremities of 200-nt fragments. Only Newbler was unable to assemble this dataset, for the same reason as in the dataset SpSim.

emoveOpenAssembler Ray produced no incorrect contigs, only 1 mismatch, and 0 indel (Tables 3 and S2). Even if the number of contigs is very low, the number of mismatches and the number of indels must also be minimized to facilitate downstream analyses. It is very important that the assembly contains the lowest number of errors because otherwise mismatches and indels will be interpreted as mutations by biologists. Given the null number of chimeric contigs for emoveOpenAssembler Ray, and the large number of mismatches for other assemblers, Ray outperformed the other assemblers for this dataset.

### 6.12.5 Real mixed datasets

Real datasets are described in Tables 1 and S1. Table 4 presents the added value provided by mixing reads, for three different genomes: *E. coli* K-12 MG1655, *A. baylyi* ADP1, and *C. curtum* DSM 15641. Comparisons with other assemblers are presented in Table S3 (*E. coli* K-12 MG1655), Table S4 (*A. baylyi* ADP1), and Table S5 (*C. curtum* DSM 15641).

***E. coli*** For *E. coli* K-12 MG1655, Newbler gave 874 contigs whereas Ray produced 126 with only Illumina reads, and 109 with both technologies (Roche/454 and Illumina) with 1 misassembled contig that aligned in two blocks: 1-83124 with 4210996-4294119 and 82863-112667 with 4294197-4323999. This contig was presumably correct according to the colinearity between aligned segments. For this genome, the Illumina reads were grouped in two paired libraries, $d_\mu$ and $d_\sigma$ had values of 215 and 11 for the first library and 486 and 26 for the second.

***A. baylyi*** For *A. baylyi* ADP1, Newbler produced 109 contigs with 380 indels. Ray assembled the Illumina reads in 259 contigs and the 454/Roche and Illumina reads in 91 contigs, which is less than 109, the number for Newbler with Roche/454 reads only. Mixed reads allowed Ray to detect homopolymer errors (indels): Newbler produced 380 and Ray only 1. For the *A. baylyi* ADP1 mixed data, the misassembled contig aligned in two blocks: 1358-560 with 2804111-2804909 and 234-1 with 2804911-2805144 – a single indel in the contig.

***C. curtum*** For *C. curtum* DSM 15641, Newbler produced 30 contigs with 8 indels, and Ray assembled data in 76 contigs (Illumina reads) and in 27 contigs (Roche/454 and Illumina reads). Ray (27 contigs) outperformed Newbler (30 contigs), and the other metrics were constant, except for the largest contig length, for which Newbler was better.

Accordingly, for these reported results, Ray is an excellent assembler – and the only one to assemble successfully mixed datasets (see Tables S3-S5). Also, Ray is better than Velvet and ABySS on Illumina data.

Furthermore, the assembly of *E. coli* K-12 MG1655 with paired short sequences confirmed that read length does not matter for de novo assembly – a pair of reads can be transformed explicitely in a longer sequence of a length $d$ ($d$ being near the $d_\mu$ of its corresponding paired library) like in EULER-SR [9], or implicitely, like in Ray.

### 6.12.6 Scalability

Ray assembles reads in an efficient way. In Tables 3, 4, and S2-S5, the running time indicates that Ray ranked well in speed evaluation. Ray is scalable because it uses messages passing, like ABySS. Scalability will be instrumental, if not paramount, to tackle not only larger genomes, but also to sort out the tremendous amount of data and weed out the errors coming from the cutting-edge sequencers, such as the Illumina HiSeq 2000 – which has a reported output of 150-200 Gb. To the best of our knowledge, only ABySS and Ray make extensive use of message passing interface, allowing them to run on many computers at once.

## 6.13 Conclusion

The throughput of new sequencing instruments is overwhelming assemblers. Thus, better bioinformatics systems are necessary and critical in sequence assembly. The cost of DNA sequencing has dramatically lowered over the last years while the sheer volume of data generated by sequencing instruments has grown exponentially The lower cost makes genome research more affordable whereas the generated volume of data increases the complexity of the analyses. To assist in these analyses, numerous assembly algorithms have been published over the last years. However, none of these can tackle simultaneously a mix of reads from different sequencing systems. The ability to assemble simultaneously such mixes of reads is useful to alleviate the error rates of each

single sequencing platform. Ray allows this type of analysis. In principle, genomes obtained with Ray will require less sequence finishing.

Our contributions are an open source parallel assembler, the best results on simulated datasets (Tables 3 and S2), improved assemblies on Illumina reads in comparison with current state-of-the-art short-read assemblers (Tables 3 and S3-S5), and the first assembler that can simultaneously assemble reads from a mix of sequencing technologies (Tables 4 and S3-S5).

Future directions are to assemble a human genome with Ray, and to adapt the code for the new upcoming sequencing platforms. With new sequencing technologies emerging, an assembler that can handle all of them in an efficient fashion is highly desirable.

## 6.14 Acknowledgments

We thank the Canadian Institutes of Health Research (JC) and the Natural Sciences and Engineering Research Council of Canada (FL: NSERC discovery grant 262067 ) for research funding. JC holds the Canada Research Chair in Medical Genomics. SB is recipient of a Master's award (200902CGM-204212-172830) and a Doctoral award (200910GSD-226209-172830) from the Canadian Institutes of Health Research. We also wish to acknowledge the CLUMEQ consortium and Compute Canada for access to computing infrastructure (colosse.clumeq.ca), and the Canadian Foundation for Innovation for infrastructure funding (ls30.genome.ulaval.ca). We are grateful to Mohak Shah, Frédéric Raymond, Ken Dewar, Elénie Godzaridis, and Éric Paquet for critical reading of the manuscript. We are greatly indebted to Dr. Torsten Seemann for providing precious feedbacks.

## 6.15 Author disclosure statement

No competing financial interests exist.

## 6.16 References

[1] Altschul, S. et al. 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* 25, 3389–3402.

[2] Aury, J.-M. et al. 2008. High quality draft sequences for prokaryotic genomes using a mix of new sequencing technologies. *BMC Genomics* 9, 603.

[3] Barbe, V. et al. 2004. Unique features revealed by the genome sequence of Acinetobacter sp. ADP1, a versatile and naturally transformation competent bacterium. *Nucleic Acids Res* 32, 5766–5779.

[4] Batzoglou, S. et al. 2002. Arachne: a whole-genome shotgun assembler. *Genome Res* 12, 177–189.

[5] Bentley, D. R. et al. 2008. Accurate whole human genome sequencing using reversible terminator chemistry. *Nature* 456, 53–59.

[6] Blattner, F. R. et al. 1997. The Complete Genome Sequence of Escherichia coli K-12. *Science* 277, 1453–1462.

[7] Butler, J. et al. 2008. ALLPATHS: de novo assembly of whole-genome shotgun microreads. *Genome Res* 18, 810–820.

[8] Chaisson, M. et al. 2004. Fragment assembly with short reads. *Bioinformatics* 20, 2067–2074.

[9] Chaisson, M. J. et al. 2008. De novo fragment assembly with short mate-paired reads: Does the read length matter? *Genome Res* 19, 336–346.

[10] Chaisson, M. J. et al. 2008. Short read fragment assembly of bacterial genomes. *Genome Res* 18, 324–330.

[11] Diguistini, S. et al. 2009. De novo genome sequence assembly of a filamentous fungus using Sanger, 454 and Illumina sequence data. *Genome Biol* 10, R94.

[12] Dohm, J. C. et al. 2007. SHARGCS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Res* 17, 1697–1706.

[13] Flicek, P. and Birney, E. 2009. Sense from sequence reads: methods for alignment and assembly. *Nat Methods* 6, S6-S12.

[14] Gallant, J. K. 1983. The complexity of the overlap method for sequencing biopolymers. *J Theor Biol* 101, 1–17.

[15] Goldberg, S. M. D. et al. 2006. A Sanger/pyrosequencing hybrid approach for the generation of high-quality draft assemblies of marine microbial genomes. *Proc Natl Acad Sci U S A* 103, 11240–11245.

[16] Hernandez, D. et al. 2008. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome Res* 18, 802–809.

[17] Holt, R. A. and Jones, S. J. M. 2008. The new paradigm of flow cell sequencing. *Genome Res* 18, 839–846.

[18] Hoskins, J. et al. 2001. Genome of the Bacterium Streptococcus pneumoniae Strain R6. *J Bacteriol* 183, 5709–5717.

[19] Hossain, M. S. et al. 2009. Crystallizing short-read assemblies around seeds. *BMC Bioinformatics* 10 Suppl 1, S16.

[20] Hutchinson, G. 1969. Evaluation of polymer sequence fragment data using graph theory. *Bull Math Biophys* 31, 541–562.

[21] Idury, R. M. and Waterman, M. S. 1995. A new algorithm for DNA sequence assembly. *J Comput Biol* 2, 291–306.

[22] Jeck, W. R. et al. 2007. Extending assembly of short DNA sequences to handle error. *Bioinformatics* 23, 2942–2944.

[23] Kurtz, S. et al. 2004. Versatile and open software for comparing large genomes. *Genome Biol* 5, R12.

[24] Margulies, M. et al. 2005. Genome sequencing in microfabricated high-density picolitre reactors. *Nature* 437, 376–380.

[25] Mavromatis, K. et al. 2009. Complete genome sequence of Cryptobacterium curtum type strain (12-3T). *Stand Genomic Sci* 1, 96–100.

[26] Medini, D. et al. 2008. Microbiology in the post-genomic era. *Nat Rev Microbiol* 6, 419–430.

[27] Miller, J. R. et al. 2008. Aggressive assembly of pyrosequencing reads with mates. *Bioinformatics* 24, 2818–2824.

[28] Myers, E. W. et al. 2000. A whole-genome assembly of Drosophila. *Science* 287, 2196–2204.

[29] Pevzner, P. A. et al. 2001. An Eulerian walk approach to DNA fragment assembly. *Proc Natl Acad Sci U S A* 98, 9748–9753.

[30] Pop, M. 2009. Genome assembly reborn: recent computational challenges. *Brief Bioinform* 10, 354–366.

[31] Rothberg, J. M. and Leamon, J. H. 2008. The development and impact of 454 sequencing. *Nat Biotechnol* 26, 1117–1124.

[32] Sanger, F. et al. 1977. DNA sequencing with chain-terminating inhibitors. *Proc Natl Acad Sci U S A* 74, 5463–5467.

[33] Scheibye-Alsing, K. et al. 2009. Sequence assembly. *Comput Biol Chem* 33, 121–136.

[34] Simpson, J. T. et al. 2009. ABySS: A parallel assembler for short read sequence data. *Genome Res* 19, 1117–1123.

[35] Sleator, D. D. and Tarjan, R. E. 1985. Self-adjusting binary search trees. *Journal of the ACM* 32, 652–686.

[36] Smith, L. M. et al. 1986. Fluorescence detection in automated DNA sequence analysis. *Nature* 321, 674–679.

[37] Warren, R. L. et al. 2007. Assembling millions of short DNA sequences using SSAKE. *Bioinformatics* 23, 500–501.

[38] Zerbino, D. R. and Birney, E. 2008. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res* 18, 821–829.

## 6.17 Tables

Table 6.1: Dataset descriptions.

| Dataset | Description | Reads accession key |
|---|---|---|
| **Simulations – *S. pneumoniae* R6** | | |
| SpSim | simulated 50-bp reads, depth of coverage: 50 | - |
| SpErSim | SpSim with 1% mismatch | - |
| SpPairedSim | simulated 200-bp fragments, 50-bp reads representing their extremities, depth of coverage: 50 | - |
| **Mixed dataset 1 – *E. coli* K12 MG1655** | | |
| Roche/454 | Roche/454 reads | sra/SRA001028 |
| Illumina | Illumina paired reads | sra/SRA001125 |
| Mixed | - | sra/SRA001125, sra/SRA001028 |
| **Mixed dataset 2 – *A. baylyi* ADP1** | | |
| Roche/454 | Roche/454 reads | sra/SRA003611 |
| Illumina | Illumina unpaired reads | sra/SRA003611 |
| Mixed | - | sra/SRA003611 |
| **Mixed dataset 3 – *C. curtum* DSM 15641** | | |
| Roche/454 | Roche/454 reads | sra/SRA008863 |
| Illumina | Illumina unpaired reads | sra/SRA008863 |
| Mixed | - | sra/SRA008863 |

Table 6.2: Assemblers.

| Assembler | Version |
|---|---|
| ABySS [276] | 1.1.2 |
| EULER-SR [9] | 1.1.2 |
| Newbler [24] | 2.0.00.20 |
| Ray | 0.0.7 |
| Velvet [307] | 0.7.61 |

Table 6.3: Assemblies of simulated error-free and error-prone datasets.

| Assembler | Contig ≥ 500 bp | Bases (bp) | Mean size (bp) | N50 (bp) | Largest contig (bp) | Genome coverage (%) | Incorrect contigs | Mismatches | Indels | Running time |
|---|---|---|---|---|---|---|---|---|---|---|
| **SpSim** | | | | | | | | | | |
| ABySS | 417 | 1898819 | 4553 | 7349 | 27222 | 0.9343 | 0 | 4 | 0 | 1m56.066s |
| EULER-SR | 261 | 1967594 | 7538 | 11621 | 61396 | 0.9419 | 6 | 68 | 123 | 7m22.779s |
| Velvet | 280 | 1917129 | 6846 | 11279 | 44362 | 0.9437 | 1 | 23 | 8 | 2m15.931s |
| Ray | 259 | 1954999 | 7548 | 11561 | 77867 | 0.9608 | 0 | 0 | 0 | 3m25.240s |
| **SpErSim** | | | | | | | | | | |
| ABySS | 418 | 1898547 | 4541 | 7349 | 27222 | 0.9342 | 0 | 4 | 0 | 4m52.727s |
| EULER-SR | 267 | 1965104 | 7359 | 11477 | 61349 | 0.9413 | 6 | 79 | 237 | 11m15.383s |
| Velvet | 290 | 1913682 | 6598 | 10302 | 42572 | 0.9423 | 2 | 27 | 11 | 2m40.792s |
| Ray | 259 | 1939235 | 7487 | 11554 | 77853 | 0.9531 | 0 | 0 | 0 | 4m29.223s |
| **SpPairedSim** | | | | | | | | | | |
| ABySS | 151 | 2019778 | 13376 | 22045 | 104182 | 0.9815 | 0 | 213 | 9 | 3m38.944s |
| EULER-SR | 235 | 1976831 | 8412 | 12383 | 61593 | 0.9458 | 13 | 69 | 187 | 9m59.464s |
| Velvet | 113 | 1950222 | 17258 | 32111 | 123292 | 0.9565 | 30 | 382 | 140 | 2m15.371s |
| Ray | 96 | 1964569 | 20464 | 36692 | 127906 | 0.9632 | 0 | 1 | 0 | 5m52.834s |

Table 6.4: Assemblies of mixed readouts. Roche/454 reads were assembled with Newbler whereas Illumina and mixed data were assembled with Ray.

| Data | Contig ≥ 500 bp | Bases (bp) | Mean size (bp) | N50 (bp) | Largest contig (bp) | Genome coverage (%) | Incorrect contigs | Mismatches | Indels | Running time |
|---|---|---|---|---|---|---|---|---|---|---|
| **Mixed dataset 1: *E. coli* K-12 MG1655** | | | | | | | | | | |
| Illumina | 126 | 4591168 | 36437 | 72499 | 174569 | 0.9818 | 0 | 2 | 4 | 47m54.377s |
| Roche/454 | 874 | 4513335 | 5163 | 8771 | 42344 | 0.9731 | 9 | 64 | 247 | 29m53.841s |
| Mixed | 109 | 4579657 | 42015 | 87318 | 268385 | 0.9831 | 1 | 234 | 6 | 62m30.978s |
| **Mixed dataset 2 – *A. baylyi* ADP1** | | | | | | | | | | |
| Illumina | 259 | 3677696 | 14199 | 25852 | 72730 | 0.9749 | 0 | 82 | 6 | 29m48.993s |
| Roche/454 | 109 | 3547847 | 32549 | 61793 | 214173 | 0.9846 | 0 | 69 | 380 | 43m3.785s |
| Mixed | 91 | 3540404 | 38905 | 82891 | 215819 | 0.9804 | 1 | 7 | 1 | 36m27.635s |
| **Mixed dataset 3 – *C. curtum* DSM 15641** | | | | | | | | | | |
| Illumina | 72 | 1606647 | 22314 | 36518 | 91303 | 0.9862 | 0 | 1 | 1 | 19m51.388s |
| Roche/454 | 30 | 1609423 | 53647 | 261125 | 477358 | 0.9904 | 0 | 0 | 8 | 21m24.064s |
| Mixed | 27 | 1602133 | 59338 | 116274 | 236544 | 0.9897 | 0 | 0 | 1 | 35m8.569s |

## 6.18 Figures

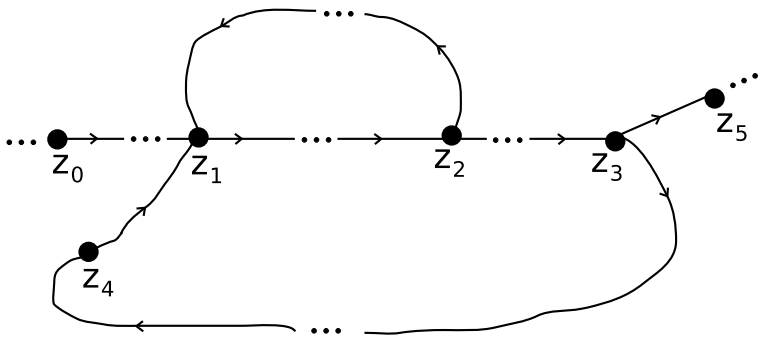


Figure 6.1: A subgraph of a de Bruijn graph.

This figure shows a part of a de Bruijn graph. In this example, short reads are not enough for the assembly problem. Suppose that the true genome sequence is of the form $\langle \cdots z_0 \cdots z_1 \cdots z_2 \cdots z_1 \cdots z_2 \cdots z_3 \cdots z_4 \cdots z_1 \cdots z_2 \cdots z_3 \cdots z_5 \cdots \rangle$. If the length of the reads (or paired reads) is smaller than the $z_1 \cdots z_2$ subsequence, no hints will help an assembly algorithm to differentiate the true sequence from the following one $\langle \cdots z_0 \cdots z_1 \cdots z_2 \cdots z_3 \cdots z_4 \cdots z_1 \cdots z_2 \cdots z_1 \cdots z_2 \cdots z_3 \cdots z_5 \cdots \rangle$. On the other hand, if there is a read that starts before $z_1$ and ends after $z_2$, there will be a possibility to solve this branching problem.

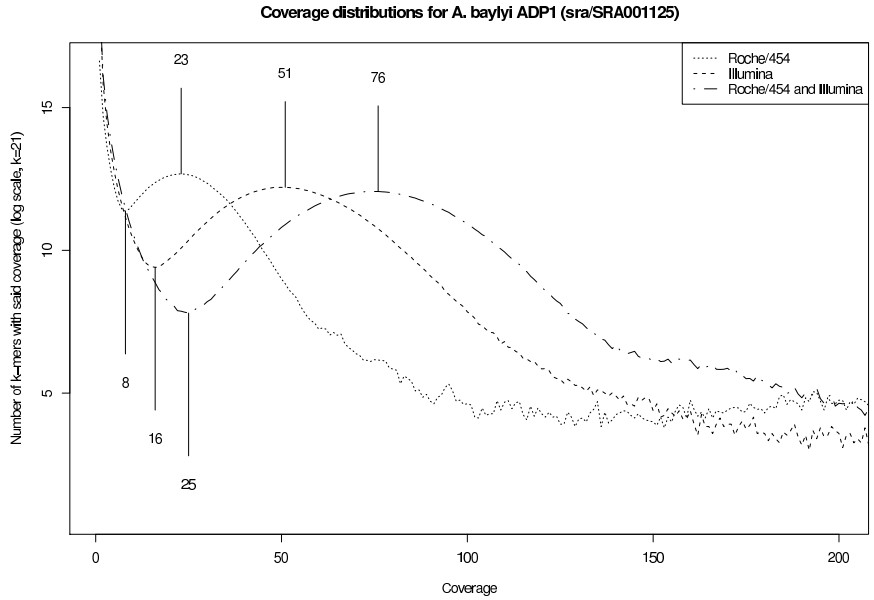


Figure 6.2: Coverage distributions.

This figure shows the coverage distributions of $k$-mers for the *A. baylyi* ADP1 dataset with Roche/454, Illumina, and Roche/454 and Illumina, $k = 21$. The minimum coverage and the peak coverage are identified for the Roche/454, Illumina , and Roche/454 and Illumina coverage distributions. The peak coverage of Roche/454+Illumina is greater than the sum of the peak coverage of Roche/454 and the peak coverage of Illumina, which suggests that the mixed approach allows one to recover low-coverage regions.

GrowSeed($w = \langle x_1, \ldots, x_l \rangle$)

$s \leftarrow 0$

WHILE $s <$ number of vertices in walk $w$ DO

$s \leftarrow$ number of vertices in walk $w$

Consider the arcs $\langle x_l, y_1 \rangle, \ldots, \langle x_l, y_m \rangle$

IF one of the $y$'s wins against all the others according to Rule 1 THEN

Extend the value of the walk $w$ by adding this particular arc

ELSE IF one of the $y$'s wins against all the others according to Rule 2 THEN

Extend the value of the walk $w$ by adding this particular arc

RETURN extended $w$

Ray(D,k,c)

Build Bruijn($D+, k, c$)

FOR each seed $w = \langle x_1, \ldots, x_l \rangle$ DO

IF the seed was not previously reached THEN

$w' \leftarrow$GrowSeed($w$)

$x \leftarrow$ the reverse-complement sequence of $w'$

$x' \leftarrow$ GrowSeed($x$)

store the walk $x'$

Merge overlapping walks in a reverse-complement fashion

RETURN the resulting contigs

Figure 6.3: The Ray algorithm.

Ray is a greedy algorithm on a de Bruijn graph. The extension of seeds is carried out by the subroutine GrowSeed. Each seed is extended using the set of Rules 1, 2, and 3. Afterwards, each extended seed is extended in the opposite direction using the reverse-complement path of the extended seed. Given two seeds $s_1$ and $s_2$, the reachability of $s_1$ from $s_2$ is not a necessary and sufficient condition of the reachability of $s_2$ from $s_1$. Owing to this property of reachability between seeds, a final merging step is necessary to remove things appearing twice in the assembly.

# Chapitre 7

# Article 3 : Ray Meta : scalable de novo metagenome assembly and profiling

## 7.1 Journal

Ce manuscrit a été publié dans le journal *Genome Biology* le 2012-12-22 pendant mes études doctorales. 

## 7.2 Résumé

Les jeux de données parallèles, particulièrement ceux provenant d'expériences de méta-génomique, nécessitent du calcul distribué pour l'assemblage *de novo* ou le profilage taxonomique. *Ray Meta* permet de faire de l'assemblage *de novo* distribué de méta-génomes et est couplé à *Ray Communities* qui permet de faire des profils dans des microbiomes en utilisant les k-mers colorés. Le logiciel peut précisément assembler et quantifier un échantillon de méta-génomique contenant 3 milliards de séquences représentant 1000 génomes bactériens présents en proportions inégales en 15 heures en utilisant 1024 cœurs de processeur, et ce en consommant seulement 1.5 Gigaoctet par cœur. Le logiciel facilitera le traitement de jeux de données complexes et énormes, et aidera à générer des pistes d'idées biologiques pour des environnements. Ray Meta est un logiciel libre et est disponible sur http ://denovoassembler.sf.net.

## 7.3 Contributions

J'ai écrit le brouillon du manuscrit, implémenté les méthodes, obtenu les collections de données, fait les simulations et les analyses. J'ai analysé les résultats avec Jacques Corbeil et Frédéric Raymond. J'ai inventé les algorithmes d'assemblage avec François Laviolette et Jacques Corbeil. Frédéric Raymond et moi avons conçu les algorithmes et stratégies de coloriage. Élénie Godzaridis et moi avons conçu les plans pour la création des logiciels parallèles et distribués.

La section 7.10 contient les contributions des auteurs.

# Ray Meta: scalable de novo metagenome assembly and profiling

Sébastien Boisvert*[1,2] Frédéric Raymond[1,2] Élénie Godzaridis[2]
François Laviolette[3] Jacques Corbeil[1,4]

2012-12-22

(1)Infectious Diseases Research Center, CHUQ Research Center, 2705, boul. Laurier, Québec (Québec), G1V 4G2, Canada.
(2)Faculty of Medicine, Laval University, 1050, av. de la Médecine, Québec (Québec), G1V 0A6, Canada.
(3)Department of Computer Science and Software Engineering, Faculty of Science and Engineering, Laval University, 1065, av. de la Médecine, Québec (Québec), G1V 0A6, Canada.
(4)Department of Molecular Medicine, Faculty of Medicine, Laval University, 1050, av. de la Médecine, Québec (Québec), G1V 0A6, Canada.

**Abstract**

Voluminous parallel sequencing datasets, especially metagenomic experiments, require distributed computing for de novo assembly and taxonomic profiling. Ray Meta is a massively distributed metagenome assembler that is coupled with Ray Communities, which profiles microbiomes based on uniquely-colored k-mers. It can accurately assemble and profile a three billion read metagenomic experiment representing 1,000 bacterial genomes of uneven proportions in 15 hours with 1,024 processor cores, using only 1.5 GB per core. The software will facilitate the processing of large and complex datasets, and will help in generating biological insights on specific environments. Ray Meta is open source and available at http://denovoassembler.sf.net.

## Keywords

metagenomics; message passing; scalability; de novo assembly; profiling; next-generation sequencing; parallel; distributed;

## 7.4 Background

While voluminous datasets from high-throughput sequencing experiments have allowed new biological questions to emerge[300, 37], the technology's speed and scalability are yet unmatched by available analysis techniques and the gap between them has been steadily growing [183, 176]. The de Bruijn graph is a structure for storing DNA words – or k-mers – that occur in sequence datasets [58, 85]. Recent work showed that adding colors to a de Bruijn graph can allow variants to be called even in the absence of a complete genome reference[126].

The field of metagenomics concerns itself with the analysis of communities by sampling the DNA of all species in a given microbial community. The assembly of metagenomes poses greater and more complex challenges than single-genome assembly as the relative abundances of the species in a microbiome are not uniform[190]. A compounding factor is the genetic diversity represented by polymorphisms and homologies between strains, which increases the difficulty of the problem for assemblers[190]. Moreover, the underlying diversity of the sample increases its complexity and adds to the difficulties of assembly. Last but not least, DNA repeats can produce misassemblies [260] in the absence of fine-tuned, accurate computational tools[10].

The microbial diversity in microbiomes contains the promise of finding new genes with novel and interesting biological functions[11]. While the throughput in metagenomics is increasing fast, bottlenecks in the analyses are becoming more apparent[269], indicating that only equally parallel – and perhaps highly distributed – analysis systems can help bridge the scalability gap. Parallel sequencing requires parallel processing for bioprospection and for making sense of otherwise largely unknown sequences.

Environmental microbiomes have been the object of several large-scale investigations. Viral genome assemblies were obtained from samples taken from hot springs[268]. Metabolic profiling of microbial communities from Antarctica[292] and Arctic [291] provided novel insights into the ecology of these communities. Furthermore, a new Archaea lineage was discovered in a hypersaline environment by means of metagenomic assembly[203]. The metabolic capabilities of terrestrial and marine microbial communities were compared[287]. The structure of communities in the environment was reconstructed[288]. All these studies brought to light that environmental microbiomes are reservoirs of genetic novelty[204], which bioprospection aims at discovering.

Through metagenomic analysis, the interplay between host and commensal microbial

metabolic activity can be studied, promising to shed light on its role in maintaining human health. Furthermore, precisely profiling the human microbial and viral flora at different taxonomic levels as well as functional profiling may hint at improved new therapeutic options[54]. To that end, the human distal gut microbiome of two healthy adults was analyzed by DNA sequencing[95], and later the human gut microbiome of 124 European individuals was analyzed by DNA sequencing from fecal samples by the MetaHIT consortium [233]. Another study proposed that there are 3 stable, location-independent, gut microbiome enterotypes [11]. Finally, the structure, function and diversity of the healthy human microbiome was investigated by the Human Microbiome Project Consortium [59].

With 16S rRNA gene sequencing, species representation can be extracted by taxonomic profiling [267]. However, using more than one marker gene produces better taxonomic profiles[165, 271]. Furthermore, a taxonomy based on phylogenetic analyses helps in the process of taxonomic profiling[182]. While taxonomic profiles are informative, functional profiling are also required to understand the biology of a system. To that end, gene ontology[12] can assign normalized functions to data.

Although not designed for metagenomes, distributed software for single genomes, such as ABySS [276] and Ray[30], illustrate how leveraging high-performance and parallel computing could greatly speed up the analysis of the large bulks of data generated by metagenome projects. Notably, sophisticated parallel tools are easily deployed on cloud computing infrastructures [265] or on national computing infrastructures through their use of a cross-platform, scalable method called message-passing interface.

Taxonomic profiling methods utilize alignments [165, 122, 187, 71, 42, 271] or hidden Markov models [141] or both [35]. Few methods are available for metagenome *de novo* assembly (MetaVelvet [202], Meta-IDBA [214], Genovo [149]), none couples taxonomic and ontology profiling with *de novo* assembly, and none is distributed to provide scalability. Furthermore, none of the existing methods for *de novo* metagenome assembly distributes the memory utilization on more than one compute machine. This additional difficulty plagues current metagenome assembly approaches.

The field of metagenomic urgently needs distributed and scalable processing methods to tackle efficiently the size of samples and the assembly and profiling challenges that this poses. Herein we show that Ray Meta, a distributed processing application is well suited for metagenomics.

We present results obtained by *de novo* metagenome assembly with coupled profiling. With Ray Meta, we show that the method scales on 2 metagenomes simulated to incorporate sequencing errors: a 100-genome metagenome assembled from $400 \times 10^6$ 101-nucleotide reads and a 1,000-genome metagenome assembled from $3 \times 10^9$ 100-nucleotide reads. Ray Communities utilizes bacterial genomes to color the assembled de Bruijn graph. The Greengenes taxonomy[182] was utilized to obtain the profiles from colored k-mers. Other taxonomies, such as the NCBI taxonomy, can be substituted readily. We also present results obtained by *de novo* metagenome assembly and taxonomic and functional profiling of 124 gut microbiomes. We compared Ray Meta to MetaVelvet and validated Ray Communities with MetaPhlAn taxonomic profiles.

## 7.5 Results

### 7.5.1 Scalability

In order to assess the scalability of Ray Meta, we simulated two large datasets. Although a simulation does not capture all genetic variations (and associated complexity) occurring in natural microbial populations, it is a mean to validate the correctness of assemblies produced by Ray Meta and the abundances predicted by Ray Communities. The first contained $400 \times 10^6$ reads, with 1% as human contamination. The remaining reads were distributed across 100 bacterial genomes selected randomly from GenBank. The read length was 101 nucleotides, the substitution error rate was 0.25% and reads were paired. Finally, the proportion of bacterial genomes followed a power law (exponent: -0.5) to mimic what is found in nature (see Materials and methods). The number of reads for this 100-genome metagenome roughly corresponds to the number of reads generated by 1 lane of a Illumina HiSeq 2000 flow cell (Illumina, Inc.). Additional file 1, Table S1 lists the number of reads for each bacterial genome and for the human genome. This dataset was assembled by Ray Meta using 128 processor cores in 13 hours, 26 minutes, with an average memory usage of 2 GB per core. The resulting assembly contained 22,162 contigs with at least 100 nucleotides and had a N50 of 152,891. The sum of contig lengths was 345,945,478 nucleotides. This is 93% of the sum of bacterial genome lengths, which was 371,623,377 nucleotides. Therefore, on average there were 3,459,454 assembled nucleotides and 221 contigs per bacterial genome, assuming that bacterial genomes were roughly of the same size and same complexity and that the coverage depth was not sufficient to assemble incorporated human contaminations. Using the known reference sequences, we validated the assembly using MUMmer to assess the

quality. There were 11,220 contigs with at least 500 nucleotides. Among these, 152 had misassemblies (1.35%). Any contig that did not align as one single maximum unique match with a breadth of coverage of at least 98.0% was marked as misassembled. The number of mismatches was 1108 while the number of insertions or deletions was 597.

To further investigate the scalability of our approach for *de novo* metagenome assembly, we simulated a second metagenome. This one contained 1,000 bacterial genomes randomly selected from GenBank as well as 1% of human sequence contamination. The proportion of the 1,000 bacterial genomes was distributed according to a power law (exponent: -0.3) and the number of reads was $3 \times 10^9$ (Additional file 1, Table S2). This number of reads is currently generated by 1 Illumina HiSeq 2000 flow cell (Illumina, Inc.). This second dataset, which is larger, was assembled *de novo* by Ray Meta in 15 hours, 46 minutes using 1,024 processor cores with an average memory usage of 1.5 GB per core. It contained 974,249 contigs with at least 100 nucleotides, a N50 of 76,095 and the sum of the contig lengths was 2,894,058,833, or 80% of the sum of bacterial genome lengths (3,578,300,288 nucleotides). Assuming a uniform distribution of assembled bases and contigs and that human sequence coverage depth was not sufficient for its *de novo* assembly, there were, on average, 974 contigs and 2,894,058 nucleotides per bacterial genomes. To validate whether or not the produced contigs were of good quality, we compared them to the known references. There were 196,809 contigs with at least 500 nucleotides. 2,638 were misassembled (1.34%) according to a very stringent test. There were 59,856 mismatches and 13,122 insertions or deletions.

Next, we sought to quantify the breadth of assembly for the bacterial genomes in the 1,000-genome dataset. In other words, the assembled percentage was calculated for each genome present in the 1,000-genome metagenome. Many of these bacterial genomes had a breadth of coverage (in the *de novo* assembly) greater than 95% (Fig. 7.1).

### 7.5.2 Estimating bacterial proportions

Another problem that can be approached with de Bruijn graphs is estimating genome nucleotide proportion within a metagenome. Using Ray Communities, the 100-genome and 1,000-genome datasets *de novo* assembled de Bruijn graphs were colored using all sequenced bacterial genomes (Additional file 1, Table S4) in order to identify contigs and to estimate bacterial proportions in the datasets. Ray Communities estimates proportions by demultiplexing k-mer coverage depth in the distributed de Bruijn graph (see Demultiplexing signals from similar bacterial strains in Materials and methods).

Because coloring occurs after *de novo* assembly has completed, the reference sequences are not needed for assembling metagenomes.

For the 100-genome dataset, only 2 bacterial genome proportions were not estimated correctly. The first was due to a duplicate in GenBank and the second to 2 almost identical genomes (Fig. 7.2A). When 2 identical genomes are provided as a basis to color the de Bruijn graph, no k-mer is uniquely colored for any of these two genomes, and identifying k-mers cannot be found through demultiplexing. This can be solved by using a taxonomy, which allows reference genomes to be similar or identical.

In the 1,000-genome dataset, 4 bacterial genome proportions were over-estimated and 20 were under-estimated (Fig. 7.2B). In both the 100-genome and 1,000-genome datasets, the proportion of bacterial genomes with incorrect estimates was 2.0%. In both of these, the incorrect estimates were caused by either duplicated genomes, identical genomes or highly similar genomes. The use of a taxonomy alleviates this problem.

The results with the 100-genome and 1,000-genome datasets shows that our method can recover bacterial genome proportions when the genome sequences are known. In real microbiome systems, there is a sizable proportion of unknown bacterial species. For this reason, it is important to devise a system that can also accomodate unknown species by using a taxonomy, which allows the classification to occurs at higher levels – such as phylum or genus instead of species.

### 7.5.3 Metagenome *de novo* assembly of real datasets

Here, we present results for 124 fecal samples from a previous study [233]. From the 124 samples, 85 were from Denmark (all annotated as being healthy) and 39 were from Spain (14 were healthy; 21 had ulcerative colitis & 4 had Crohn's disease). Each metagenome was assembled independently (Additional file 1, Table S3) and the resulting distributed de Bruijn graphs were colored to obtain taxonomic and gene ontology profiles (see Materials and methods; Additional file 1, Table S4).

These samples contained paired 75-nucleotide and/or 44-nucleotide reads obtained with Illumina Genome Analyzer sequencers. 122 samples were assembled (and profiled) in about 5 hours using 32 processor cores and the 2 remaining samples, namely MH0012 and MH0014, were assembled (and profiled) with 48 and 40 processor cores, respectively (Additional file 1, Table S3). These runtime figures include *de novo* assembly, graph coloring, signal demultiplexing and taxonomic and gene ontology profiling, which are

all tightly coupled in the process. In the next section, taxonomic profiling are presented for these 124 gut microbiome samples.

### 7.5.4 Taxonomic profiling

In metagenomic projects, the bacterial genomes that are occurring in the sample can be unknown at the species level. But it is possible to profile these samples nonetheless using a taxonomy. The key concept is to classify colored k-mers in a taxonomy tree: a k-mer is moved to a higher taxon as long as many taxons have the k-mer in order to classify it on the nearest common ancestor of the taxons. For example if not classified at the species level, it can be classified at the genus level and so on. Furthermore, taxonomy profiling does not suffer from similarity issues as seen for proportions present in samples because k-mers can be classified to higher taxons when necessary.

Accordingly, k-mers shared by several bacterial species can not be assigned to one of them accurately. To this end, the Greengenes taxonomy [182] (version 2011_11) was utilized to classify each colored k-mer in a single taxon with its taxonomic rank being one of the following: kingdom, phylum, class, order, family, genus or species. For each sample, abundances were computed at each taxonomic rank. At the moment, the most recent and accurate taxonomy for profiling taxons in a metagenome is Greengenes [182]. We profiled taxons in the 124 gut microbiome samples using this taxonomy. We also incorporated the human genome to this taxonomy to profile human abundance in the process. At the phylum level, the two most abundant taxons were *Firmicutes* and *Bacteroidetes* (Fig. 7.3A). The profile of the phylum *Chordata* indicated that 2 samples contained significantly more human sequences than the average (Fig. 7.3A). The most abundant genera in the 124 samples were *Bacteroides* and *Prevotella* (Fig. 7.3B). The taxon *Bacteroides* is reported more than once because several taxons had this name with a different ancestry in the Greengenes taxonomy. The genera *Prevotella* and *Butyrivibrio* had numerous samples with higher counts, indicating that the data is bi-modal (Fig. 7.3B). The genus *Homo* had 2 samples with significantly more abundance (Fig. 7.3B).

### 7.5.5 Grouping abundance profiles

The composition of the human gut microbiome of any individual has been proposed to be classified in one of the three enterotypes [11]. We profiled genera for each of the 124 gut microbiome samples to reproduce these three enterotypes. The 124 samples (85

from Denmark & 39 from Spain) were analyzed using the two most important principal components (Fig. 7.4; see Materials and methods). Two clear clusters are visible, one enriched for the genus *Bacteroides* and one for the genus *Prevotella*. The continuum between two enterotypes has also been reported recently [303].

### 7.5.6 Profiling of ontology terms

Gene ontology is a hierarchical classification of normalized terms in three independant domains: biological process, cellular component & molecular function. Some biological datasets are annotated with gene ontology. Here, we used gene ontology to profile the 124 metagenome samples based on a distributed colored de Bruijn graph (see Materials and methods). First, abundances for biological process terms were obtained (Fig. 7.5A). The 2 most abundant terms were metabolic process and transport. The terms oxidation-reduction process and DNA recombination had numerous sample outliers, which indicates that these samples had different biological complexity for these terms (Fig. 7.5A). Next, we sought to profile cellular component terms in the samples. The most abundant term was membrane, followed by cytoplasm, integral to membrane and plasma membrane. This redundancy is due to the hierarchical structure of gene ontology (Fig. 7.5B). Finally, we measured the abundance for molecular function terms. The most abundant was ATP binding, which had no outlier. The term DNA binding was also abundant. However, the later had outlier samples (Fig. 7.5C).

## Comparison of assemblies

Three samples from the MetaHIT Consortium [233] – MH0006 (ERS006497), MH0012 (ERS006494) and MH0047 (ERS006592) – and three samples from the Human Microbiome Project Consortium [59] (SRS011098, SRS017227, SRS018661) were assembled with MetaVelvet [202] and Ray Meta to draw a comparison. Assembly metrics are displayed in Table 7.1. The average length is higher for MetaVelvet for samples ERS006494 and ERS006592. For other samples, the average length is higher for Ray Meta. The N50 length is higher for Ray Meta for all samples. For all samples but ERS006497, the total length is higher for Ray Meta. Although we assembled the 124 samples from [233] and 313 samples (out of 764) from the Human Microbiome Project [59] with Ray Meta on supercomputers composed of nodes with little memory (24 GB), we only assembled a few samples with MetaVelvet because a single MetaVelvet assembly requires exclusive access to a single computer with large amounts of memory available (at least 128 GB). Ray Meta produced longer contigs and more bases for these 6 samples. The shared

content of assemblies produced by MetaVelvet and Ray Meta is shown in Table 7.1. A majority of assembled sequences by MetaVelvet and Ray Meta are shared. As metagenomic experiments will undoubtely become more complex, Ray Meta will gain a distinct advantage owing to its distributed implementation.

### Validation of taxonomic profiling

We compared Ray Communities to MetaPhlAn in order to validate our methodology. Taxonomic profiles for 313 samples (Additional file 2) from the Human Microbiome Project [59] were generated with Ray Communities and compared to those of MetaPhlAn [271]. Correlation is shown in Table 7.2 for various body sites. Correlations are high – for instance the correlations for buccal mucosa (46 samples) were 0.99, 0.98, 0.97, 0.98, 0.95, 0.91 for the ranks phylum, class, order, family, genus and species, respectively. These results indicate that Ray Communities has an accuracy similar to those of MetaPhlAn [271], which was utilised by the Human Microbiome Project Consortium [59]. The correlation at the genus rank for the site anterior nares was poor (0.59) because MetaPhlAn classified a high number of reads in the genus *Propionibacterium* hereby yielding a very high abundance while the number of k-mer observations classified this way by Ray Communities was more moderate. For the body site called stool, the correlation at the family rank was weak (0.62) because MetaPhlAn utilizes the NCBI taxonomy whereas Ray Communities utilizes the Greengenes taxonomy, which was shown to be more accurate [182]. Overall, these results indicate that Ray Communities yields accurate taxonomic abundances using a colored de Bruijn graph.

## 7.6 Discussion

### Message passing

Ray Meta is a method for scalable distributed *de novo* metagenome assembly whereas MetaVelvet runs only on a single computer. Therefore, fetching data with MetaVelvet is fast because only memory accesses occur. On the other hand, Ray Meta runs on many computers. Although this is a benefit at first sight, using many computers require messages to be sent back and forth in order to fetch data. Here, we used 8 nodes totalling 64 processor cores (8 processor cores per node) for Human Microbiome Project samples and the observed point-to-point latency (within our application, not the hardware latency) was around 37 microseconds – this is much more than the 100

nanoseconds required for main memory accesses. However, by minimizing messages, Ray Meta runs in an acceptable time and has a scalability unmatched by MetaVelvet while providing superior assemblies (Table 7.1).

## From Ray to Ray Meta

For single genomes, a peak coverage is required by Ray in the k-mer coverage distribution [30]. It is not the case in Ray Meta. Moreover, in Ray for single genomes, read markers are selected using the peak coverage and minimum coverage. This process is local to each read path in Ray Meta. This is in theory less precise because there are fewer coverage values, but in practice it works well as shown in this work. In Ray for single genomes, the unique k-mer coverage for a seed path (similar to a unitig) is simply the peak k-mer coverage for the whole graph whereas in Ray Meta the coverage values are sampled from the seed path only.

## Algorithms for *metagenome assembly*

Notwithstanding the non-scalability of all *de novo* metagenome assemblers except Ray Meta (MetaVelvet [202], Meta-IDBA [214], Genovo [149]), there are major differences in the algorithms these software tools implement, which are unrelated to scalability.
Genovo is an assembler for 454 reads and uses a generative probabilistic model and applies a series of hill-climbing steps iteratively until convergence [149]. For Genovo, the largest dataset processed had 311,000 reads. Herein, the largest dataset had 3,000,000,000 reads. MetaVelvet and Meta-IDBA both partition the de Bruijn subgraph using k-mer coverage peaks in the k-mer coverage distribution and/or connected components. This process does not work well in theory when there is no peak in the coverage distributions. MetaVelvet and Meta-IDBA both simplifies the de Bruijn graph iteratively – this approach, termed equivalent transformations, was introduced by Pevzner and collaborators [219]. One of the many advantages of using equivalent transformations is that the assembled sequences grow in length and their number decreases as the algorithm makes its way toward the final equivalent transformation. Equivalent transformations are hard to port to a distributed paradigm because the approach requires a mutable graph.

Ray Meta does not modify the de Bruijn subgraph in order to generate the assembly. We showed that applying a heuristics-guided graph traversal yields excellent assemblies. Furthermore, working with k-mers and their relationships directly is more amenable to

distributed computing because unlike k-mers, contigs are not regular nor small and are hard to load balance on numerous processes.

### Taxonomic profiling with k-mers

For taxonomic profiling, we have shown that Ray Communities is accurate when compared to MetaPhlAn (Table 7.2). Our approach consists in building a de Bruijn graph from the raw sequencing reads, assembling it *de novo*, and then coloring it with thousands of bacterial genomes in order to obtain an accurate profile of the sequenced metagenome. By using whole genomes instead of a few selected marker genes, such as the 16S RNA gene, some biases are removed (like the copy number of a gene). Furthermore, amplifications in a whole-genome sequencing protocol are not targeted toward any particular marker genes, which may remove further biases. A limitation of the method presented here is that using k-mers alone to compare sequences is highly stringent. On the other hand, aligner-based approaches can accommodate for an identity as low as 70% between sequences as sequence reads are usually mapped to reference bacterial genomes. At the crux of our method is the use of uniquely-colored k-mers for signal demultiplexing (see Materials and methods). Sequencing errors produce erroneous k-mers. One of the advantages of using a de Bruijn graph is that erroneous k-mers have a small probability to be considered in the assembly [30], hence sequencing errors don't contribute to taxonomic profiling for assembled sequences. However, alignment-based approaches have likely a higher sensitivity than k-mer based approaches because they are more tolerant to mismatches. Yet, the present work showed that metagenome profiling is efficiently done with k-mer counting, through the use of a colored de Bruijn graph [126], and that it is also sensitive (Fig. 7.2) and produces results similar to those of MetaPhlAn (Table 7.2). With this approach, conserved DNA regions captured the biological abundance of bacteria in a sample. A k-mer length of 31 was used to have a high stringency in the coloring process. The low error rate of the sequencing technology enabled the capture of error-free k-mers for most of the genomic regions, meaning that it was unlikely that a given k-mer was occurring in the sequence reads, in a known genome, but not in the actual sample.

## 7.6.1 Validation of assemblies

Using MUMmer [143], we validated the quality of assemblies produced by Ray Meta. The quality test used was very stringent because any contig not aligning as one single maximum unique match with a breadth of coverage of at least 98% was marked as

misassembled. In Table 7.1, the number of shared $k$-mers between assemblies produced by MetaVelvet and Ray Meta is shown. Although the overlap is significant, the $k$-mers unique to MetaVelvet or Ray Meta may be due to nucleotide mismatches. Moreover, improvements in sequencing technologies will provide longer reads with higher coverage depths. These advances will further improve *de novo* assemblies.

## 7.7 Conclusions

Scalability is a requirement for analyzing large metagenome datasets. We described a new method to assemble (Ray Meta) and profile (Ray Communities) a metagenome in a distributed fashion to provide unmatched scalability. It computes a metagenome *de novo* assembly in parallel with a de Bruijn graph. The method also yields taxonomic profiles by coloring the graph with known references and by looking for uniquely colored k-mers to identify taxons at low taxonomic ranks or by using the lowest common ancestor otherwise. Ray Meta surpassed MetaVelvet [202] for *de novo* assemblies and Ray Communities compared favorably to MetaPhlAn [271] for taxonomic profiling.

While taxonomic and functional profiling remains a useful approach to obtain a big picture of a particular sample, only *de novo* metagenome assembly can truly enable discovery of otherwise unknown genes or other important DNA sequences hidden in the data.

## 7.8 Materials and methods

Thorough documentation and associated scripts to reproduce our studies are available in Additional file 3 on the publisher website or on https://github.com/sebhtml/Paper-Replication-2012

### 7.8.1 Memory model

Ray Meta uses the message passing interface. As such, a 1,024-core job has 1,024 processes running on many computers. In the experiments, each node had 8 processor cores and 24 GB, or 3 GB per core. With the message passing paradigm, each core has its own virtual memory that is protected from any other process. Because data is distributed uniformly using a distributed hash table (DHT), memory usage for a single process is very low. For the 1,024-core job, the maximum memory usage of any process was on average 1.5 GB.

### 7.8.2 Assemblies

Metagenome assemblies with profiling were computed with Ray v2.0.0 (Additional file 4) on colosse, a Compute Canada resource. Ray is an open source software – the license is the GNU General Public License, version 3 (GPLv3) – that is freely available from http://denovoassembler.sourceforge.net/ or http://github.com/sebhtml/ray. Ray can be deployed on public compute infrastructure or in the cloud (see [263] for a review).

The algorithms implemented in the software Ray were heavily modified for metagenome *de novo* assembly and these changes were called Ray Meta. Namely, the coverage distribution for k-mers in the de Bruijn graph is not utilized to infer the average coverage depth for unique genomic regions. Instead, this value is derived from local coverage distributions during the parallel assembly process. Therefore, unlike MetaVelvet [202], Ray Meta does not attempt to calculate or use any global k-mer coverage depth distribution.

### 7.8.3 Simulated metagenomes with a power law

Two metagenomes (100 and 1,000 genomes, respectively) were simulated with abundances following a power law (Additional file 1, Tables S1 & S2). Power law is commonly found in biological systems [18]. Simulated sequencing errors were randomly distributed and the error rate was valued at 0.25% and the average insert length was 400. The second simulated metagenome was assembled with 128 8-core computers (1,024 processor cores) interconnected with a Mellanox ConnectX QDR Infiniband fabric (Mellanox, Inc.). For the 1,000-genome dataset, messages were routed with a de Bruijn graph of degree 32 and diameter 2 to reduce the latency.

### 7.8.4 Validation of assemblies

Assembled contigs were aligned onto reference genomes using the MUMmer bioinformatics software suite [143]. More precisely, deltas were generated with nucmer. Using show-coords, any contig not aligning as one single maximum with at least 98% breadth of coverage was marked as misassembled. Contigs aligning in two parts at the beginning and end of a reference were not counted as misassembled owing to the circular nature of bacterial genomes. Finally, small insertions/deletions and mismatches were obtained with show-snps.

### 7.8.5 Colored and distributed de Bruijn graphs

The vertices of the de Bruijn graph are distributed across processes called ranks. Here, graph coloring means labeling of the vertices of a graph. A different color is added to the graph for each reference sequence. Each k-mer in any reference sequence is colored with the reference sequence color if it occurs in the distributed de Bruijn graph. Therefore, any k-mer in the graph has 0, 1 or more colors. First, a k-mer with 0 color indicates that the k-mer does not exist in provided databases. Second, a k-mer with 1 color means that this k-mer is specific to one and only one reference genome in the provided databases while at least 2 colors indicates that the k-mer is not specific to one single reference sequence. These reference sequences are assigned to leaves in a taxonomic tree. Reference sequences can be grouped in independent namespaces. Genome assembly is independent from graph coloring.

### 7.8.6 Demultiplexing signals from similar bacterial strains

Biological abundances were estimated by using the product of the number of k-mer matched in the distributed de Bruijn graph by the mode coverage of k-mers that were uniquely colored. This number is called the k-mer observations. The total of k-mer observations is the sum of coverage depth values of all colored k-mers. A proportion is calculated by dividing the k-mer observations by the total.

### 7.8.7 Taxonomic profiling

All bacterial genomes available in GenBank [25] were utilized for coloring the distributed de Bruijn graphs (Additional file 1, Table S4). Each k-mer was assigned to a taxon in the taxonomic tree. When a k-mer has different taxon colors, the coverage depth was assigned to the nearest common ancestor.

### 7.8.8 Gene ontology profiling

The de Bruijn graph was colored with coding sequences from the EMBL nucleotide sequence database [142] (EMBL_CDS), which are mapped to gene ontology by transitivity using the uniprot mapping to gene ontology [41]. For each ontology term, coverage depths of colored k-mers were added to obtain its total number of k-mer observations.

### 7.8.9 Principal component analysis

Principal component analysis was used to group taxonomic profiles to produce enterotypes. Data were prepared in a matrix using the genera as rows and the samples as columns. Singular values and left and right singular vectors of this matrix were obtained using singular value decomposition implemented in R. The right singular vectors were sorted by singular values. The sorted right singular vectors were used as the new base for the re-representation of the genus proportions. The two first dimensions were plotted.

### 7.8.10 Software implementation

Ray Meta is a distributed software that runs on connected computers by transmitting messages over a network using the message-passing interface (MPI) and is implemented in C++. The MPI standard is implemented in libraries such as Open-MPI [91] and MPICH2 [101]. On each processor core, tasks are divided in smaller ones and given to a pool of 32768 workers (thread pool), which are similar to chares in CHARM++ [134]. Each of these sends messages to a virtual communicator. The later implements a message aggregation strategy in which messages are automatically multiplexed and de-multiplexed. The $k$-mers are stored in a distributed sparse hash table which utilizes open addressing (double hashing) for collisions. Incremental resizing is utilized in this hash table when the occupancy exceeds 90% to grow tables locally. Smart pointers are utilized in this table to perform real-time memory compaction. The software is implemented on top of RayPlatform, a development framework to ease the creation of massively distributed high performance computing applications.

## Comparison with MetaVelvet

Software versions were MetaVelvet 1.2.01, velvet 1.2.07, and Ray 2.0.0 (with Ray Meta). MetaVelvet was run on one processor core. Ray Meta was run on 64 processor cores for Human Microbiome Project samples (SRS011098, SRS017227, SRS018661) and on 48, 32, and 32 processor cores for ERS006494, ERS006497, and ERS006592 (MetaHIT samples), respectively. There were 8 processor cores per node. The running time for MetaVelvet is the sum of running times for velveth, velvetg and meta-velvetg. For MetaVelvet, sequence files were filtered to remove any sequence with more than 10 N symbols. The resulting files were shuffled to create files with interleaved sequences. The insert size was manually provided to MetaVelvet and the k-mer length was set to

51 as suggested in its documentation. Peak coverages were determined automatically by MetaVelvet. Ray Meta was run with a k-mer length of 31. No other parameters were required for Ray Meta and sequence files were provided without modification to Ray Meta. The overlaps of assemblies produced by MetaVelvet and by Ray Meta were evaluated with Ray using the graph coloring features. No mismatches were allowed in k-mers. Overlaps were computed for scaffolds with at least 500 nucleotides.

### Comparison with MetaPhlAn

Taxonomic profiles calculated with MetaPhlAn [271] for samples from the Human Microbiome Project were obtained [59]. Taxonomic profiles were produced by Ray Communities for 313 samples (Additional file 2). Pearson's correlation was calculated for each body site by combining taxon proportions for both methods for each taxonomic rank.

## 7.9 Competing interests

The authors declare that they have no competing interests.

## 7.10 Authors' contributions

SB drafted the manuscript, implemented methods, gathered public data and performed simulations and analyses. SB, JC and FR analyzed results. SB, FL and JC designed *de novo* assembly algorithms. SB and FR designed graph coloring strategies. EG and SB devised parallel distributed software designs. All authors read and approved the final manuscript.

## 7.11 Acknowledgements

Computations were performed on the Colosse supercomputer at Université Laval and the Guillimin supercomputer at McGill University (resource allocation project: nne-790-ab), under the auspices of Calcul Québec and Compute Canada. The operations of Guillimin and Colosse are funded by the Canada Foundation for Innovation (CFI), the National Science and Engineering Research Council (NSERC), NanoQuébec, and the Fonds Québécois de Recherche sur la Nature et les Technologies (FQRNT). Tests

were also carried out on the Mammouth-parallèle II super computer at Université de Sherbrooke (Réseau Québécois de calcul de haute performance, RQCHP).

JC is the Canada Research Chair in Medical Genomics. SB is recipient of a Frederick Banting and Charles Best Canada Graduate Scholarship Doctoral Award (200910GSD-226209-172830) from the Canadian Institutes for Health Research (CIHR). FR and JC acknowledge the support of the Consortium Québécois sur la découverte du médicament (CQDM) and of Mitacs through the Mitacs-Accelerate program. This research was supported in part by the Fonds de recherche du Québec - Nature et technologies (grant 2013-PR-166708 to FL and JC) and by the Discovery Grants Program (Individual, Team and Subatomic Physics Project) from the Natural Sciences and Engineering Research Council of Canada (grant 262067 to FL).

## 7.12 References

[1] Wold B, Myers RM: **Sequence census methods for functional genomics**. *Nature Methods* 2008, **5**:19–21.

[2] Brenner S: **Sequences and consequences**. *Philosophical Transactions of the Royal Society B: Biological Sciences* 2010, **365**:207–212.

[3] McPherson JD: **Next-generation gap**. *Nature Methods* 2009, **6**:S2–S5.

[4] Mardis E: **The $1,000 genome, the $100,000 analysis?** *Genome Medicine* 2010, **2**:84+.

[5] Compeau PEC, Pevzner PA, Tesler G: **How to apply de Bruijn graphs to genome assembly**. *Nature Biotechnology* 2011, **29**:987–991.

[6] Flicek P, Birney E: **Sense from sequence reads: methods for alignment and assembly**. *Nature Methods* 2009, **6**:S6–S12.

[7] Iqbal Z, Caccamo M, Turner I, Flicek P, McVean G: **De novo assembly and genotyping of variants using colored de Bruijn graphs**. *Nature Genetics* 2012, **44**:226–232.

[8] Miller JR, Koren S, Sutton G: **Assembly algorithms for next-generation sequencing data.** *Genomics* 2010, **95**:315–327.

[9] Salzberg SL: **Beware of mis-assembled genomes**. *Bioinformatics* 2005, **21**:4320–4321.

[10] Treangen TJ, Salzberg SL: **Repetitive DNA and next-generation sequencing: computational challenges and solutions**. *Nature Reviews Genetics* 2011, **13**:36–46.

[11] Lorenz P, Eck J: **Metagenomics and industrial applications**. *Nature Reviews Microbiology* 2005, **3**:510–516.

[12] Scholz MB, Lo CC, Chain PSG: **Next generation sequencing and bioinformatic bottlenecks: the current state of metagenomic data analysis**. *Current Opinion in Biotechnology* 2012, **23**:9–15.

[13] Schoenfeld T, Patterson M, Richardson PM, Wommack KE, Young M, Mead D: **Assembly of Viral Metagenomes from Yellowstone Hot Springs**. *Applied and Environmental Microbiology* 2008, **74**:4164–4174.

[14] Varin T, Lovejoy C, Jungblut AD, Vincent WF, Corbeil J: **Metagenomic Analysis of Stress Genes in Microbial Mat Communities from Antarctica and the High Arctic**. *Applied and Environmental Microbiology* 2012, **78**:549–559.

[15] Varin T, Lovejoy C, Jungblut AD, Vincent WF, Corbeil J: **Metagenomic profiling of Arctic microbial mat communities as nutrient scavenging and recycling systems**. *Limnology and Oceanography* 2010, **55**:1901–1911.

[16] Narasingarao P, Podell S, Ugalde JA, Brochier-Armanet C, Emerson JB, Brocks JJ, Heidelberg KB, Banfield JF, Allen EE: **De novo metagenomic assembly reveals abundant novel major lineage of Archaea in hypersaline microbial communities**. *The ISME Journal* 2011, **6**:81–93.

[17] Tringe SG, von Mering C, Kobayashi A, Salamov AA, Chen K, Chang HW, Podar M, Short JM, Mathur EJ, Detter JC, Bork P, Hugenholtz P, Rubin EM: **Comparative Metagenomics of Microbial Communities**. *Science* 2005, **308**:554–557.

[18] Tyson GW, Chapman J, Hugenholtz P, Allen EE, Ram RJ, Richardson PM, Solovyev VV, Rubin EM, Rokhsar DS, Banfield JF: **Community structure and metabolism through reconstruction of microbial genomes from the environment**. *Nature* 2004, **428**:37–43.

[19] Naviaux RK, Good B, McPherson JD, Steffen DL, Markusic D, Ransom B, Corbeil J: **Sand DNA - a genetic library of life at the water's edge**. *Marine Ecology Progress Series* 2005, **301**:9–22.

[20] Cho I, Blaser MJ: **The human microbiome: at the interface of health and disease**. *Nature Reviews Genetics* 2012, **13**:260–270.

[21] Gill SR, Pop M, Deboy RT, Eckburg PB, Turnbaugh PJ, Samuel BS, Gordon JI, Relman DA, Fraser-Liggett CM, Nelson KE: **Metagenomic Analysis of the Human Distal Gut Microbiome**. *Science* 2006, **312**:1355–1359.

[22] Qin J, Li R, Raes J, Arumugam M, Burgdorf KS, Manichanh C, Nielsen T, Pons N, Levenez F, Yamada T, Mende DR, Li J, Xu J, Li S, Li D, Cao J, Wang B, Liang H, Zheng H, Xie Y, Tap J, Lepage P, Bertalan M, Batto JM, Hansen T, Le Paslier D, Linneberg A, Nielsen HB, Pelletier E, Renault P, Sicheritz-Ponten T, Turner K, Zhu H, Yu C, Li S, Jian M, Zhou Y, Li Y, Zhang X, Li S, Qin N, Yang H, Wang J, Brunak S, Dore J, Guarner F, Kristiansen K, Pedersen O, Parkhill J, Weissenbach J, Bork P, Ehrlich SD, Wang J: **A human gut microbial gene catalogue established by metagenomic sequencing**. *Nature* 2010, **464**:59–65.

[23] Arumugam M, Raes J, Pelletier E, Le Paslier D, Yamada T, Mende DR, Fernandes GR, Tap J, Bruls T, Batto JMM, Bertalan M, Borruel N, Casellas F, Fernandez L, Gautier L, Hansen T, Hattori M, Hayashi T, Kleerebezem M, Kurokawa K, Leclerc M, Levenez F, Manichanh C, Nielsen HB, Nielsen T, Pons N, Poulain J, Qin J, Sicheritz-Ponten T, Tims S, Torrents D, Ugarte E, Zoetendal EG, Wang J, Guarner F, Pedersen O, de Vos WM, Brunak S, Doré J, MetaHIT Consortium, Antolín M, Artiguenave F, Blottiere HM, Almeida M, Brechot C, Cara C, Chervaux C, Cultrone A, Delorme C, Denariaz G, Dervyn R, Foerstner KU, Friss C, van de Guchte M, Guedon E, Haimet F, Huber W, van Hylckama-Vlieg J, Jamet A, Juste C, Kaci G, Knol J, Lakhdari O, Layec S, Le Roux K, Maguin E, Mérieux A, Melo Minardi R, M'rini C, Muller J, Oozeer R, Parkhill J, Renault P, Rescigno M, Sanchez N, Sunagawa S, Torrejon A, Turner K, Vandemeulebrouck G, Varela E, Winogradsky Y, Zeller G, Weissenbach J, Ehrlich SD, Bork P: **Enterotypes of the human gut microbiome.** *Nature* 2011, **473**:174–180.

[24] Consortium THMP: **Structure, function and diversity of the healthy human microbiome**. *Nature* 2012, **486**:207–214.

[25] Schloss PD, Handelsman J: **Introducing DOTUR, a Computer Program for Defining Operational Taxonomic Units and Estimating Species Richness**. *Applied and Environmental Microbiology* 2005, **71**:1501–1506.

[26] Liu B, Gibbons T, Ghodsi M, Pop M: **MetaPhyler: Taxonomic profiling for metagenomic sequences**. In *Bioinformatics and Biomedicine (BIBM), 2010 IEEE International Conference on*, IEEE 2010:95–100.

[27] Segata N, Waldron L, Ballarini A, Narasimhan V, Jousson O, Huttenhower C: **Metagenomic microbial community profiling using unique clade-specific marker genes**. *Nature Methods* 2012, **9**:811–814.

[28] McDonald D, Price MN, Goodrich J, Nawrocki EP, DeSantis TZ, Probst A, Andersen GL, Knight R, Hugenholtz P: **An improved Greengenes taxonomy with explicit ranks for ecological and evolutionary analyses of bacteria and archaea**. *The ISME Journal* 2011, **6**:610–618.

[29] Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, Harris MA, Hill DP, Issel-Tarver L, Kasarskis A, Lewis S, Matese JC, Richardson JE, Ringwald M, Rubin GM, Sherlock G: **Gene ontology: tool for the unification of biology. The Gene Ontology Consortium.** *Nature Genetics* 2000, **25**:25–29.

[30] Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJM, Birol I: **ABySS: A parallel assembler for short read sequence data.** *Genome Research* 2009, **19**:1117–1123.

[31] Boisvert S, Laviolette F, Corbeil J: **Ray: Simultaneous Assembly of Reads from a Mix of High-Throughput Sequencing Technologies**. *Journal of Computational Biology* 2010, **17**:1519–1533.

[32] Schatz MC, Langmead B, Salzberg SL: **Cloud computing and the DNA data race.** *Nature Biotechnology* 2010, **28**:691–693.

[33] Huson DH, Mitra S, Ruscheweyh HJ, Weber N, Schuster SC: **Integrative analysis of environmental sequences using MEGAN4**. *Genome Research* 2011, **21**:1552–1560.

[34] Meyer F, Paarmann D, D'Souza M, Olson R, Glass EM, Kubal M, Paczian T, Rodriguez A, Stevens R, Wilke A, Wilkening J, Edwards RA: **The metagenomics**

**RAST server - a public resource for the automatic phylogenetic and functional analysis of metagenomes**. *BMC Bioinformatics* 2008, **9**:386–8.

[35] Dixon P: **VEGAN, a package of R functions for community ecology**. *Journal of Vegetation Science* 2003, **14**:927–930.

[36] Caporaso JG, Kuczynski J, Stombaugh J, Bittinger K, Bushman FD, Costello EK, Fierer N, Pena AG, Goodrich JK, Gordon JI, Huttley GA, Kelley ST, Knights D, Koenig JE, Ley RE, Lozupone CA, McDonald D, Muegge BD, Pirrung M, Reeder J, Sevinsky JR, Turnbaugh PJ, Walters WA, Widmann J, Yatsunenko T, Zaneveld J, Knight R: **QIIME allows analysis of high-throughput community sequencing data**. *Nature Methods* 2010, **7**:335–336.

[37] Krause L, Diaz NN, Goesmann A, Kelley S, Nattkemper TW, Rohwer F, Edwards RA, Stoye J: **Phylogenetic classification of short environmental DNA fragments.** *Nucleic Acids Research* 2008, **36**:2230–2239.

[38] Brady A, Salzberg SL: **Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated Markov models.** *Nature Methods* 2009, **6**:673–676.

[39] Namiki T, Hachiya T, Tanaka H, Sakakibara Y: **MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads**. *Nucleic Acids Research* 2012.

[40] Peng Y, Leung HCM, Yiu SM, Chin FYL: **Meta-IDBA: a de Novo assembler for metagenomic data**. *Bioinformatics* 2011, **27**:i94–i101.

[41] Laserson J, Jojic V, Koller D: **Genovo: De Novo Assembly for Metagenomes**. *Journal of Computational Biology* 2011, **18**:429–443.

[42] Wu GD, Chen J, Hoffmann C, Bittinger K, Chen YYY, Keilbaugh SA, Bewtra M, Knights D, Walters WA, Knight R, Sinha R, Gilroy E, Gupta K, Baldassano R, Nessel L, Li H, Bushman FD, Lewis JD: **Linking long-term dietary patterns with gut microbial enterotypes.** *Science (New York, N.Y.)* 2011, **334**:105–108.

[43] Pevzner PA, Tang H, Waterman MS: **An Eulerian path approach to DNA fragment assembly.** *Proceedings of the National Academy of Sciences* 2001, **98**:9748–9753.

[44] Kurtz S, Phillippy A, Delcher AL, Smoot M, Shumway M, Antonescu C, Salzberg SL: **Versatile and open software for comparing large genomes.** *Genome Biol* 2004, **5**.

[45] Schadt EE, Linderman MD, Sorenson J, Lee L, Nolan GP: **Computational solutions to large-scale data management and analysis.** *Nature Reviews Genetics* 2010, **11**:647–657.

[46] Barabasi AL, Oltvai ZN: **Network biology: understanding the cell's functional organization**. *Nature Reviews Genetics* 2004, **5**:101–113.

[47] Benson DA, Boguski MS, Lipman DJ, Ostell J: **GenBank**. *Nucleic Acids Research* 1997, **25**:1–6.

[48] Kulikova T, Aldebert P, Althorpe N, Baker W, Bates K, Browne P, van den Broek A, Cochrane G, Duggan K, Eberhardt R, Faruque N, Garcia-Pastor M, Harte N, Kanz C, Leinonen R, Lin Q, Lombard V, Lopez R, Mancuso R, McHale M, Nardone F, Silventoinen V, Stoehr P, Stoesser G, Ann M, Tzouvara K, Vaughan R, Wu D, Zhu W, Apweiler R: **The EMBL Nucleotide Sequence Database.** *Nucleic Acids Research* 2004, **32**.

[49] Camon E, Magrane M, Barrell D, Lee V, Dimmer E, Maslen J, Binns D, Harte N, Lopez R, Apweiler R: **The Gene Ontology Annotation (GOA) Database: sharing knowledge in Uniprot with Gene Ontology.** *Nucleic Acids Research* 2004, **32**:D262–266.

[50] Gabriel E, Fagg G, Bosilca G, Angskun T, Dongarra J, Squyres J, Sahay V, Kambadur P, Barrett B, Lumsdaine A, Castain R, Daniel D, Graham R, Woodall T, Gabriel E, Fagg GE, Bosilca G, Angskun T, Dongarra JJ, Squyres JM, Sahay V, Kambadur P, Barrett B, Lumsdaine A, Castain RH, Daniel DJ, Graham RL, Woodall TS: **Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation Recent Advances in Parallel Virtual Machine and Message Passing Interface**. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, *Volume 3241 of* Lecture Notes in Computer Science. Edited by Kranzlmüller D, Kacsuk P, Dongarra J, Berlin, Heidelberg: Springer Berlin / Heidelberg 2004:353–377.

[51] Gropp W: **MPICH2: A New Start for MPI Implementations**. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, *Volume 2474 of*

Lecture Notes in Computer Science. Edited by Kranzlmüller D, Volkert J, Kacsuk P, Dongarra J, Berlin, Heidelberg: Springer Berlin / Heidelberg 2002:37–42.

[52] Kale LV, Krishnan S: **CHARM++: a portable concurrent object oriented system based on C++**. In *Proceedings of the eighth annual conference on Object-oriented programming systems, languages, and applications*, OOPSLA '93, New York, NY, USA: ACM 1993:91–108.

## 7.13 Additional files

### Additional file 1: Tables S1, S2, S3 & S4

Table S1: Composition of the simulated 100-genome metagenome.

Table S2: Composition of the simulated 1,000-genome metagenome.

Table S3: Overlay data on metagenome assembly of 124 gut microbiome samples.

Table S4: List of genomes used for coloring de Bruijn graphs.

### Additional file 2 — List of 313 samples from the Human Microbiome Project

### Additional file 3 — Documentation and scripts to reproduce all experiments

### Additional file 4 — Software source code for Ray Meta and Ray Communities

## 7.14 Figure legends

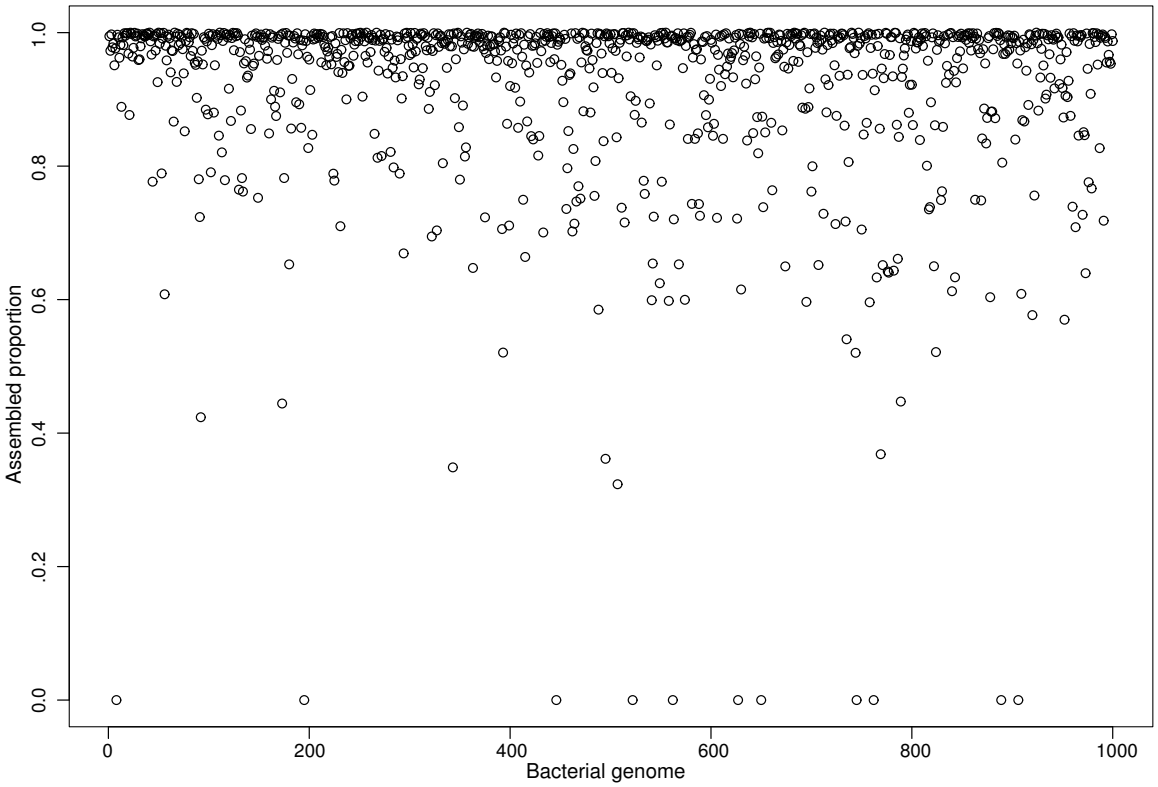


Figure 7.1: Assembled proportions of bacterial genomes for a simulated metagenome with sequencing errors.

$3 \times 10^9$ 100-nucleotide reads were simulated with sequencing errors (0.25%) from a simulated metagenome containing 1,000 bacterial genomes with proportions following a power law. Having 1,000 genomes with power law proportions makes it impossible to classify sequences with their coverage. This large metagenomic dataset was assembled using distributed de Bruijn graphs and profiled with colored de Bruijn graphs. Highly similar, but different genomes, are likely to be hard to assemble. This figure shows the proportion of each genome that was assembled *de novo* within the metagenome. 88.2% of the bacterial genomes were assembled at least with a breadth of coverage of 80.0%.

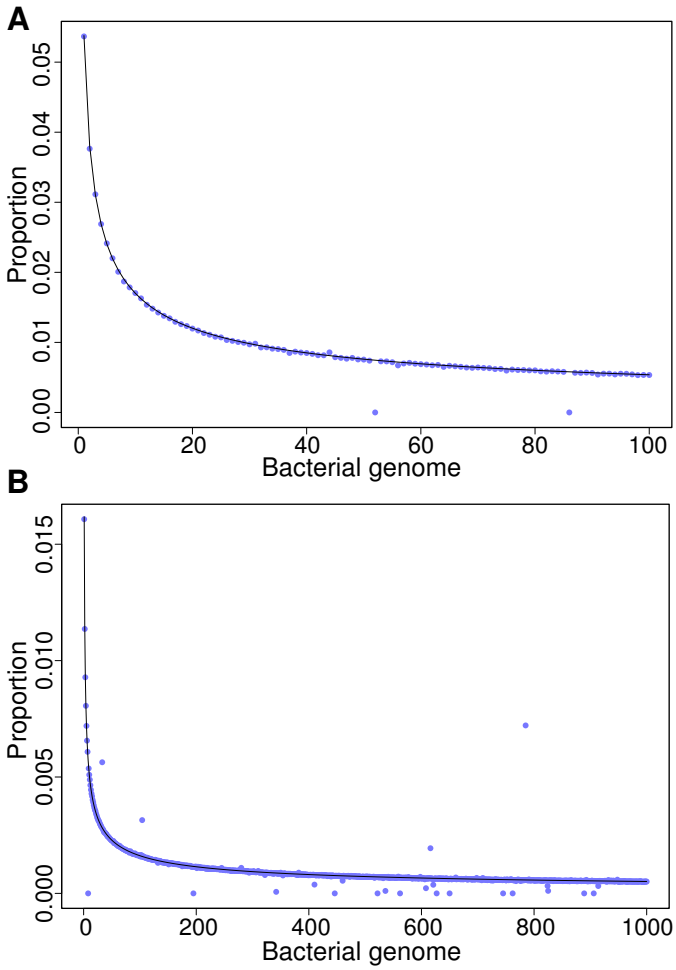


Figure 7.2: Estimated bacterial genome proportions.

For the two simulated metagenomes (100 & 1,000 bacterial genomes, respectively), colored de Bruijn graphs were utilized to estimate the nucleotide proportion of each bacterial genome in its containing metagenome. Genome proportions in metagenomes followed a power law. Black lines show expected nucleotide proportion for bacterial genomes while blue points represent proportions measured by colored de Bruijn graphs. (*A*) For the 100-genome metagenome, only 2 bacterial genomes were not correctly measured (2.0%), namely *Methanococcus maripaludis* X1 and *Serratia* AS9. *Methanococcus maripaludis* X1 was not detected because it was duplicated in the data set as *Methanococcus maripaludis* XI, thus providing 0 uniquely colored k-mers. *Serratia* AS9 was not detected because it shares almost all its k-mers with *Serratia* AS12. (*B*) For the 1,000-genome metagenome, 4 bacterial genomes were over-estimated (0.4%) while 20 were under-estimated (2.0%). These errors were due to highly similar bacterial genomes, hence not providing uniquely colored k-mers. This problem can be alleviated by either using a curated set of reference genomes or by using a taxonomy. The remaining 976 bacterial genomes had a measured proportion near the expected value.

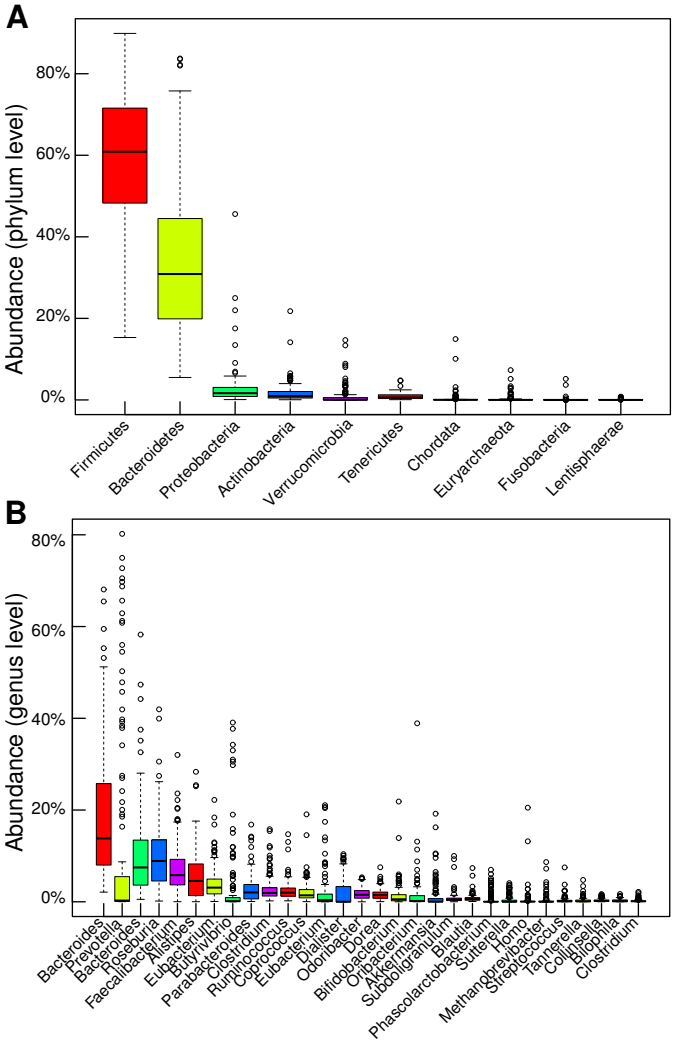


Figure 7.3: Fast and efficient taxonomic profiling with distributed colored de Bruijn graphs.

124 metagenomic samples containing short paired reads from a previous study were assembled *de novo* and profiled for taxons. The graph coloring occured once the de Bruijn graph was assembled *de novo*. (*A*) The taxonomic profiles are shown for the phylum level. The two most abundant phyla were Firmicutes and Bacteroidetes. This is in agreement with the litterature [233]. The abundance of human sequences was also measured. The phylum Chordata had two outlier samples. This indicates that 2 of the samples had more human sequences than the average, which may bias results. (*B*) At the genus level, the most abundant taxon was *Bacteroides*. This taxon occured more than once because it was present at different locations within the Greengenes taxonomic tree. Also abundant is the genus *Prevotella*. Furthermore, the later had numerous samples with higher counts, which may help in non-parametric clustering. Two samples had higher abundance of human sequences, as indicated by the abundance of the genus *Homo*.

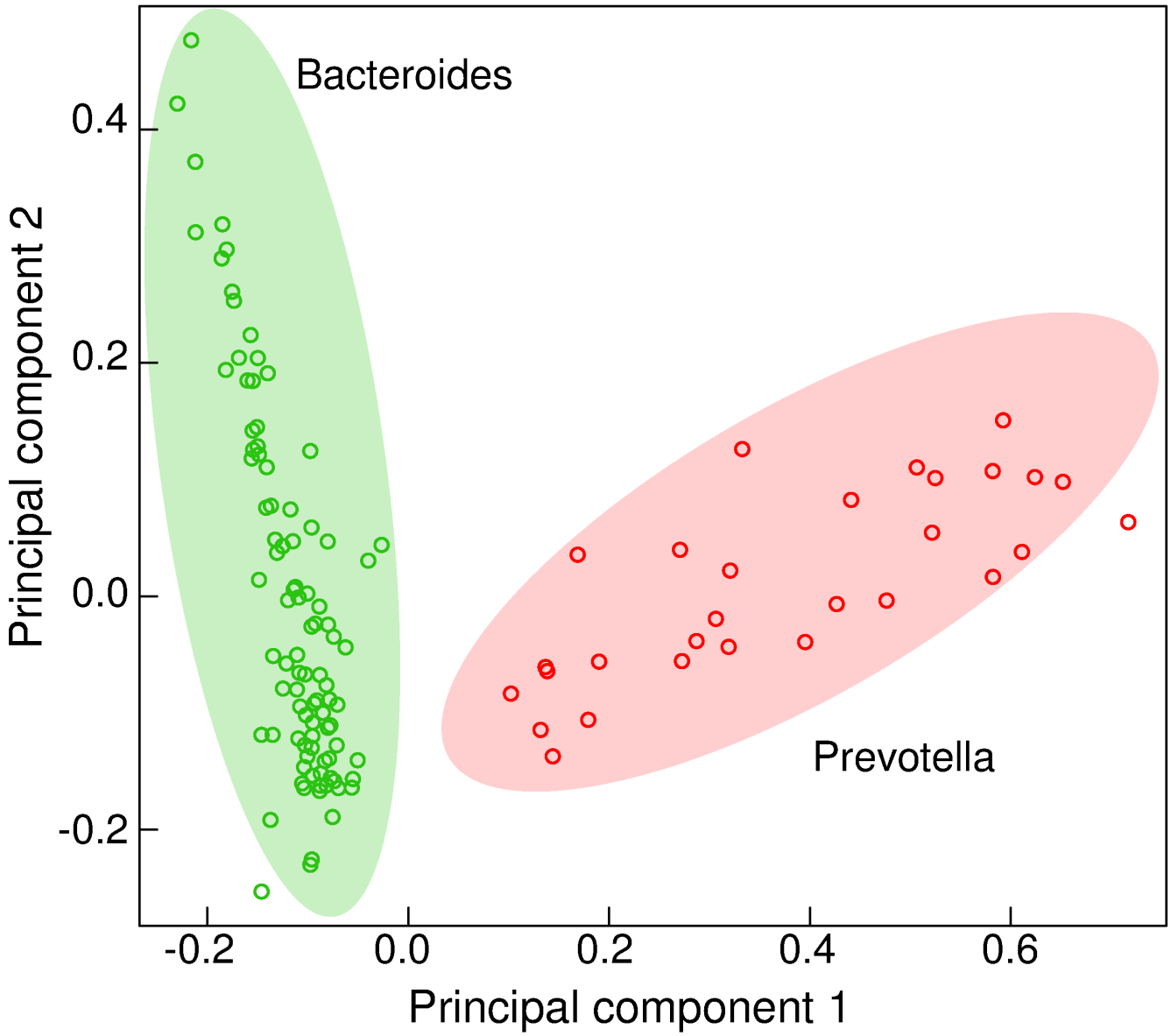


Figure 7.4: Principal component analysis shows 2 clusters.

Here, a principal component analysis (see Materials and methods) with abundances at the genus level yielded 2 distinct clusters. Abundances were obtained with colored de Bruijn graphs. One was enriched in the genus *Bacteroides* while the other was enriched in the genus *Prevotella*. The principal component 1 was linearly correlated with the genus *Prevotella* while the principal component 2 was linearly correlated with the genus *Bacteroides*. This analysis suggests that there is a continuum between the two abundant genus *Bacteroides* and *Prevotella*. This interpretation differs from the original publication in which 3 human gut enterotypes were reported [11]. More recently, it was proposed that there are only two enterotypes and individuals are distributed in a continuum between the two [303].

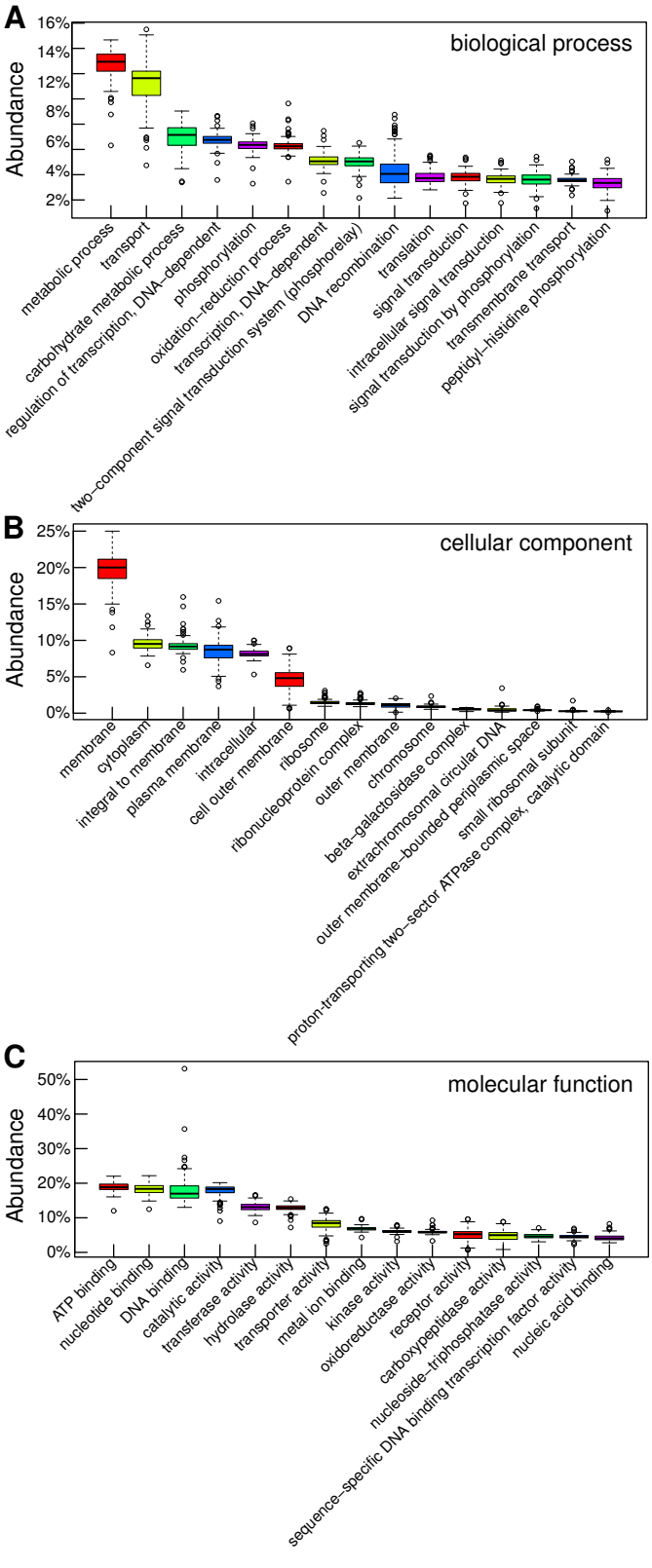
A
biological process
Abundance
16%
14%
12%
8%
6%
4%
2%
metabolic process
transport
carbohydrate metabolic process
regulation of transcription, DNA−dependent
phosphorylation
oxidation−reduction process
transcription, DNA−dependent
two−component signal transduction system (phosphorelay)
DNA recombination
translation
signal transduction
intracellular signal transduction
signal transduction by phosphorylation
transmembrane transport
peptidyl−histidine phosphorylation
B
cellular component
Abundance
25%
20%
15%
10%
5%
0%
membrane
cytoplasm
integral to membrane
plasma membrane
intracellular
cell outer membrane
ribosome
ribonucleoprotein complex
outer membrane
chromosome
beta−galactosidase complex
extrachromosomal circular DNA
outer membrane−bounded periplasmic space
proton−transporting two−sector ATPase complex, catalytic domain
small ribosomal subunit
C
molecular function
Abundance
50%
40%
30%
20%
10%
0%
ATP binding
nucleotide binding
DNA binding
catalytic activity
transferase activity
hydrolase activity
transporter activity
metal ion binding
kinase activity
oxidoreductase activity
receptor activity
carboxypeptidase activity
nucleoside−triphosphatase activity
sequence−specific DNA binding transcription factor activity
nucleic acid binding

Figure 7.5: Ontology profiling with colored de Bruijn graphs.

Gene ontology profiles were obtained by coloring of the graph resulting from *de novo* assembly. Gene ontology has three domains: biological process, cellular component & molecular function. For each domain, only the 15 most abundant terms are displayed. ($A$) Ontology terms in the biological process domain were profiled. Some of these have several outlier samples, namely oxidation-reduction process and DNA recombination. ($B$) Ontology profiling for cellular component terms is shown. The most abundant is the membrane term. ($C$) The profile for molecular function terms is shown. Binding functions are the most abundant with ATP binding, nucleotide binding and DNA binding in the top 3. Next is catalytic activity, which is a general term. More specific catalytic activities are listed.

## 7.15 Tables and captions

Table 7.1: Comparison of assemblies produced by MetaVelvet and Ray Meta.

| | MetaVelvet | Ray Meta | Shared |
|---|---|---|---|
| **ERS006494** | | | |
| Reads | 372,147,956 | | |
| Scaffolds | 50,136 | 56,363 | |
| Total length (nt) | 150,904,880 | 156,075,852 | 130,979,321 |
| Average length (nt) | 3,009 | 2,769 | |
| N50 length (nt) | 6,141 | 12,117 | |
| Longest length (nt) | 146,549 | 570,359 | |
| **ERS006497** | | | |
| Reads | 322,444,920 | | |
| Scaffolds | 61,093 | 52,194 | |
| Total length (nt) | 113,403,805 | 111,187,163 | 94,649,612 |
| Average length (nt) | 1,856 | 2,130 | |
| N50 length (nt) | 2,778 | 5,430 | |
| Longest length (nt) | 115,684 | 430,963 | |
| Running time (h:min) | 4:34 | 10:06 | |
| **ERS006592** | | | |
| Reads | 53,869,960 | | |
| Scaffolds | 4,358 | 9,387 | |
| Total length (nt) | 19,501,348 | 24,687,275 | 18,061,386 |
| Average length (nt) | 4,474 | 2,629 | |
| N50 length (nt) | 8,819 | 10,277 | |
| Longest length (nt) | 87,983 | 137,473 | |
| Running time (h:min) | 0:41 | 4:28 | |
| **SRS011098** | | | |
| Read | 202,487,723 | | |
| Scaffolds | 30,458 | 36,130 | |
| Total length (nt) | 60,574,679 | 83,736,387 | 51,938,031 |
| Average length (nt) | 1,988 | 2,317 | |
| N50 length (nt) | 3,117 | 4,961 | |
| Longest length (nt) | 192,898 | 222,213 | |
| Running time (h:min) | 8:34 | 6:38 | |
| **SRS017227** | | | |
| Reads | 139,002,751 | | |
| Scaffolds | 106,957 | 89,953 | |
| Total length (nt) | 171,200,737 | 186,958,660 | 126,068,148 |
| Average length (nt) | 1,600 | 2,078 | |
| N50 length (nt) | 2,168 | 3,771 | |
| Longest length (nt) | 102,749 | 224,709 | |
| Running time (h:min) | 9:00 | 7:10 | |
| **SRS018661** | | | |
| Reads | 288,475,194 | | |
| Scaffolds | 30,709 | 18,541 | |
| Total length (nt) | 35,281,226 | 36,891,130 | 21,659,465 |
| Average length (nt) | 1,148 | 1,989 | |
| N50 length (nt) | 1,223 | 3,794 | |
| Longest length (nt) | 111,404 | 377,149 | |
| Running time (h:min) | 1:24 | 4:42 | |

Only scaffolds with a length higher or equal to 500 were considered.

Table 7.2: Correlation of taxonomic abundances produced by MetaPhlAn and Ray Communities.

| Body site | samples | phylum | class | order | family | genus | species |
|---|---|---|---|---|---|---|---|
| anterior nares | 45 | 0.91 | 0.92 | 0.94 | 0.94 | 0.59 | 0.59 |
| attached keratinized gingiva | 3 | 0.99 | 0.94 | 0.92 | 0.94 | 0.84 | 0.71 |
| buccal mucosa | 46 | 0.99 | 0.98 | 0.97 | 0.98 | 0.95 | 0.91 |
| left retroauricular crease | 3 | 0.99 | 0.99 | 0.99 | 0.99 | 0.72 | 0.83 |
| mid vagina | 1 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.90 |
| palatine tonsils | 4 | 0.90 | 0.80 | 0.79 | 0.83 | 0.84 | 0.97 |
| posterior fornix | 23 | 0.99 | 0.99 | 0.99 | 0.99 | 0.97 | 0.94 |
| right retroauricular crease | 6 | 0.94 | 0.92 | 0.93 | 0.94 | 0.83 | 0.91 |
| saliva | 3 | 0.97 | 0.87 | 0.88 | 0.96 | 0.89 | 0.95 |
| stool | 61 | 0.80 | 0.81 | 0.81 | 0.62 | 0.92 | 0.84 |
| subgingival plaque | 5 | 0.86 | 0.75 | 0.76 | 0.74 | 0.81 | 0.93 |
| supragingival plaque | 53 | 0.94 | 0.93 | 0.92 | 0.88 | 0.89 | 0.93 |
| throat | 6 | 0.95 | 0.86 | 0.87 | 0.92 | 0.92 | 0.80 |
| tongue dorsum | 53 | 0.93 | 0.80 | 0.79 | 0.84 | 0.85 | 0.88 |
| vaginal introitus | 1 | 1.00 | 1.00 | 0.99 | 0.99 | 0.99 | 0.97 |
| Total | 313 | | | | | | |

Pearson's correlation was utilized to compare taxonomic abundance for 313 samples from various body sites [59].

# Chapitre 8

# Article 4 : Human analysts at superhuman scales : what has friendly software to do ?

## 8.1 Journal

Ce manuscrit a été accepté pour publication dans le journal *Big Data* le 18 novembre 2013. Cet article sera en accès libre.

Je suis deuxième auteur et j'ai contribué également avec la première auteure.

## 8.2 Résumé

Puisque les analystes doivent analyser de plus en plus de données en moins de temps, les créateurs de logiciels pour les grosses données sont mis à l'épreuve par le besoin d'efficacité améliorée. Ray, notre assembleur de génomes rapide et utilisable, permet de résoudre des problèmes de grosses données en utilisant des ressources optimales et en produisant une solution rapide, correcte et conservatrice. Seulement en faisant abstraction de la taille des données tant du côté des ordinateurs que des humains, la vraie question scientifique – complexe elle-même – peut éventuellement être résolue. Afin de mettre un drap par dessus les détails mécaniques de calcul des grosses données, nous avons développé RayPlatform, un cadre de développement de logiciel qui permet aux utilisateurs de se concentrer sur les problèmes spécifiques au domaine. RayPlatform est un environnement par passage de messages qui supporte les nuages, les superordi-

nateurs et les ordinateurs de bureau. En utilisant des technologies établies comme le C++ ou MPI (pour *message-passing interface*), nous supportons les génomes de plusieurs centaines d'espèces allant des virus aux plantes, en utilisant les machines allant des ordinateurs de bureau aux superordinateurs. À partir de cette expérience, nous présentons des idées pour rendre le temps des ordinateurs plus utile, et le temps des humains plus précieux.

## 8.3 Contributions

J'ai co-écrit le manuscrit, éditer le texte pour y incorporer les références, soumis le matériel à l'éditeur et répondu aux commentaires des évaluateurs. J'ai aussi réalisé les figures et les tests de mise à l'échelle du logiciel.

## 8.4 Contenu

Running title : Best practices for designing scalable tools

**Human analysts at superhuman scales : what has friendly software to do ?**

Élénie Godzaridis[1,*], Sébastien Boisvert[2,3,*], Fangfang Xia[4], Mikhail Kandel[5,6], Steve Behling[6], Bill Long[6], Carlos P. Sosa[6,7], François Laviolette[8], and Jacques Corbeil[2,3]

Corresponding author : Sébastien Boisvert

* These authors contributed equally to this work.

[1] Strategic Technology, Bentley Systems, Inc. 3365 Boulevard Ste-Anne, Beauport, Quebec G1E 3L1, Canada.

[2] Faculty of Medicine, Laval University, 1050, av. de la Médecine, Québec (Québec), G1V 0A6, Canada.

[3] Infectious and immune diseases, CHUQ Research Center, 2705, boul. Laurier, Québec (Québec), G1V 4G2, Canada.

[4] Mathematics and Computer Science Division, Argonne National Laboratory, 9700 S. Cass Ave., Lemont, Illinois, 60439, U.S.A.

[5] Department of Electrical and Computer Engineering, University of Illinois, Champaign, IL 61820, U.S.A.

[6] Cray Inc., 380 Jackson Street, Suite 210, St. Paul, MN 55101, U.S.A.

[7] Biomedical Informatics and Computational Biology (BICB), University of Minnesota Rochester, 599 Walter Library, 117 Pleasant Ave. SE, Minneapolis, MN 55455, U.S.A.

[8] Department of Computer Science and Software Engineering, Faculty of Science and Engineering, Laval University, 1065, av. de la Médecine, Québec (Québec), G1V 0A6, Canada.

elenie.godzaridis@bentley.com, sebastien.boisvert.3@ulaval.ca, fangfang@anl.gov, kandel3@illinois.edu, sbehling@cray.com, longb@cray.com, cpsosa@cray.com, francois.laviolette@ift.ulaval.ca, jacques.corbeil@crchul.ulaval.ca,

## 8.5 Abstract

As analysts are expected to process a greater amount of information in a shorter time, creators of big data software are challenged with the need for improved efficiency. Ray, our group's usable, scalable genome assembler, addresses big data problems by using optimal resources, and producing one, correct and conservative, timely solution. Only by abstracting the size of the data from both the computers and the humans can the real scientific question, often complex in itself, be eventually solved. To draw a curtain over the specific computational machinery of big data, we developed RayPlatform, a programming framework that allows users to concentrate on their domain-specific problems. RayPlatform is a parallel message-passing software framework that runs on clouds, supercomputers and desktops alike. Using established technologies such as C++ and MPI (message-passing interface), we handle the genomes of hundreds of species from viruses to plants, using machines ranging from desktop computers to supercomputers. From this experience, we present insights on making computer time more useful – and user time much more valuable.

## 8.6 Introduction

It is no secret that life sciences have made their entry in the world of big data [178, 265, 263, 280, 77, 281, 108]. Modern sequencing technologies produce terabytes of data in the span of a few days [26, 1]. Many large-scale projects have been set up to take advantage of these technologies in fields as varied as ecology [59, 278], infectious diseases [250, 232, 200, 175] and phylogeny [42, 170, 169, 171, 62, 156, 302]. Analysts have lamented about the unyielding nature of their tools [183, 176]. Sequencing the human genome might once have taken ten years, cost millions, and yielded countless appearances in mainstream media, but today it takes one week, costs approximately $10,000, and is barely worth a mention in the methods section. Yet the classes of algorithms used to piece it together are largely unchanged [217, 125, 197, 219, 190, 229, 196].

An approach used to deal with the large scale of the data produced is to similarly increase the scale of the resources devoted to it : for example using supercomputers [166]. Several genome assemblers exist that can use an MPI (message passing interface, see [87]) library to scale over thousands of computing nodes [276, 30, 128, 127, 129, 32]. Kiki is also a genome assembler that uses MPI (https ://github.com/GeneAssembly/kiki). Ray, which we developed, parallelizes every step of its execution. Notably, ABySS (a genome assembler that uses MPI for building the genome graph) was used to assemble the very large genome of white spruce [27].

The genome assembly problem is commonly represented as a graph problem [124, 92, 125, 219, 196]. With the input sequence reads, a graph is constructed. Then, paths are discovered in this graph. There are different types of graphs for genome assembly, and it was shown that the two popular ones (de Bruijn graph and unitig/overlap graph) are closely connected [196]. Genome assembly is difficult because repeats in genomes are usually longer than the readouts obtained by DNA sequencing.

The methods used in Ray to parallelize the assembly problem over as many processing elements as are available are readily applicable to most types of graph problems. For example, the approach of dividing the exploration of a graph between several actors by giving them different starting points was first reported in 2009 by Jackson and colleagues [128] using a IBM Blue Gene/L.

Here we cover not only increasing scalability through parallelization but also various constraints facing big data analysis software such as ours, and present the ways we have counteracted them. These constraints cover the costs in time and resources of running the software, of course, but also the incidental costs to the user of setting up the analysis environment, accessing the results, and choosing the parameters.

For example, one of the foremost issues to be taken into consideration when publishing scalable analysis software for large datasets is that of portability. Not everyone interested in scaling out their analysis can or will use supercomputers ; indeed, there is a growing interest in running all kinds of biological analyses on the cloud [234, 140, 20]. Ray's design has allowed it to be easily ported over to a cloud environment. As an example, Ray is available in the DNAnexus platform (https ://platform.dnanexus.com/app/ray) which allows apps to be started with a few mouse clicks. Even among supercomputers, special care has to be taken so that the software is flexible enough to maintain its performance on different types of machines such as the Cray XE6, the IBM Blue Gene/Q and university computing infrastructures. It has to be friendly enough so that the time saved for the user by the increased performance is not offset by the time cost of getting the software to run in the first place.

We also introduce Ray Cloud Browser, a web-based genome visualization tool that allows users to more closely examine and validate their results.

## 8.7 Definitions

Here, graphs are utilized for representing the genomic data and for optional communication tasks such as message routing. A directed graph $G = (V, A)$ is a set of vertices $V$ and a set of directed relationships (called arcs) $A \subseteq V \times V$. A de Bruijn graph is a directed graph defined for an alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_{|\Sigma|}\}$ and an integer $k \in \mathbb{N}$. Its vertices are the sequences of $k$ elements from the alphabet. There is an arc $(u, v) \in$ between two vertices $u$ and $v$ if and only if the last $k-1$ elements of $u$ and the $k-1$ elements of $v$ are the same. In genome research, the alphabet is $\Sigma = \{A, T, C, G\}$ (nucleotides) and the integer $k$ usually falls roughly within the range from 19 [85] and 150 – the current maximum length for the sequencing platform that provides the highest yield. In the problems that Ray aims to solve, the graph is a distributed subgraph (meaning that its vertices are not stored all in one location). The complete de Bruijn graph is not required because only the DNA sequences that occur in a given sample are relevant – any genome contains a small proportion of all possible sequences of length $k$ if $k$ is large enough. A *path* $P$ of a graph is an ordered list of vertices $\langle v_1, v_2, v_3, \ldots, v_{|P|} \rangle$ such that any two consecutive vertices $v_i$, $v_{i+1}$ are linked by an arc. Here, we define a traversal as the process by which a path is constructed. More precisely, in the context of *de novo* genome assembly, the traversal is guided by heuristics based on spatial information provided by input data [30]. Reads (usually at least 100 nucleotides) are larger than vertices (for example $k = 31$). Furthermore, heuristics also use long-range information provided by paired reads and mate pairs.

A parallel job comprises an ordered set of processes, which are called ranks. A message is a unit of information passed from one rank to another. The granularity refers to the trade-off

between the amount of communication and the amount of processing in any given period. A coarse granularity means that processing is not heavily interleaved with communication and that communication occurs once in a while. A fine granularity is characterized by a high level of overlap between communication and processing.

## 8.8 Building a message-passing framework

A significant part of the challenges involved in writing scalable, parallel scientific software are common to algorithms and applications. Because factors such as the size of messages, frequency of communications and number of potential recipients are largely dependent on the scale of the problem rather than its specific application domain, they are ideal candidates for abstraction.

RayPlatform, the framework Ray is built on (see Figure 8.1), abstracts and optimizes three different aspects of the performance of an application built on message passing, that is, non-trivially parallel : communications between the application's ranks (see Section 8.9), task switching and scheduling (see Section 8.10), and data access and retrieval (see Section 8.11). The source code of RayPlatform is licensed under the GNU Lesser General Public License (version 3) and is available at https ://github.com/sebhtml/RayPlatform. A minimalist example using RayPlatform is available at https ://github.com/sebhtml/RayPlatform-example.

Ray 2.2.0 contains 26 plugins that are registered on the RayPlatform using its application programming interface (API). In that regard, the source code of Ray focuses on solving biological problems such as *de novo* genome (or metagenome) assembly, taxonomic profiling, and other tasks [32] instead of solving framework-related problems, which are handed over to RayPlatform. The source code of Ray is licensed under the GNU General Public License (version 3) and is available at https ://github.com/sebhtml/ray.

## 8.9 Talk is not cheap

As demonstrated before [224], communication overhead is a looming specter – casting its shadow over the hopes of supercomputing scalability and performance. When tackling a problem with many nodes (motherboards), though now equipped with more memory and processing threads, it is not uncommon to witness a slowdown in computation. Misplaced data, stalled execution flow or high-latency networks can all affect the whole compute system. One strategy to reduce the number of messages that transit in an interconnect is to use message aggregation.

The best supercomputers available implement highly optimized hardware message routing : Cray XT/XE [270] and IBM Blue Gene/Q [51] use multidimensional tori while cutting-edge

Cray XC [82] and IBM PERCS [10] systems use a transparently managed dynamic hierarchy of all-to-all networks. In these cases there is dedicated proprietary hardware that ensures messages arrive in a timely and consistent manner, along with packaging expertise to understand the proper placement of computer components. However, not every computing facility is equipped with such sophisticated communication networks and some interconnects have poor performance in the presence of an any-to-any communication pattern. With an any-to-any communication pattern, any rank can send a message directly to any other rank. To counteract this effect, there are several levels of control over communication intensity and efficiency in RayPlatform. First, different kinds of software routing graphs can be used to alleviate the load on some parts of the network (see [23] for an introduction to this subject) in order to bring down the numbers of direct physical connections each rank is required to maintain. This facility can also balance the messages that transit through a rank. In RayPlatform, software message routing is a service provided by a virtual message router. In addition to all-to-all and per-node routing schemes, RayPlatform supports Kautz graphs (offering the smallest diameter), de Bruijn graphs, polytope routing [23, 22, 24] and torus graphs (modelling the physical interconnect of many supercomputers [82, 51]). On Université Laval's supercomputer, Colosse (960 compute nodes, 7680 processor cores), this yields a five times lower latency (500 to 90 microseconds) for a 512-rank job. Ray components use a tighly-coupled communication pattern. Therefore reducing the point-to-point communication latency reduces the computation time.

Messages are aggregated to get economies of scale, and ranks cannot receive more than one message per cycle. This allows ranks to use the work quantum they do each cycle to advance their own problem, rather than just answer requests for data from other ranks. This communication strategy works well when the work is balanced across ranks. In Ray, work is derived from ownerships of vertices in the de Bruijn graph. Ownership is determined by hashing a vertex, and therefore the vertices in the graph are uniformly distributed on ranks, and so is the communication. The data are stored in a in-memory distributed hash table (DHT) with double hashing [139] and incremental resizing.

RayPlatform also includes a prototype for mini-ranks to adequately leverage systems with a higher number of cores per machine but a slower interconnect between individual nodes. Mini-ranks are ranks, which is to say they send messages to each other and do their own computation. One of the differences is that mini-ranks can be pooled together inside a process to reduce the number of MPI processes doing calls to MPI functions. There is no routing between mini-ranks of the same rank and the communication is guaranteed to be in-memory, which lessens the load on the network. This hybrid programming model allows us to extend RayPlatform's paradigm to handle varying combination of network and computing performance, and to keep using the message-passing interface (MPI) paradigm to write the software. In

RayPlatform, mini-ranks are implemented as POSIX threads.

## 8.10 A question of efficiency

Because the kind of compute power required for big data applications is the kind of compute power that, very often, is only available for rental (by-the-hour usage on shared infrastructure), as is the case in clouds and academic clusters, one source of waste is paying for nodes that don't do their fair share of work. As a result, load balancing is often a prime concern, and it's also difficult to manage. Even when no data dependency exists between the many execution threads, efficiency requires splitting the data so that the varying amounts of work required by each datum add up to the same amount for each rank ; this would be a variation on the NP-hard knapsack problem [136]. In molecular dynamics, this very problem was addressed in CHARM++ using task migration where workers move between compute nodes in order to maintain a balanced load across every compute nodes [134, 135].

While traversing a path in the graph does not necessarily depend on intermediate results obtained by the computation of any other path, the problem still requires constant communication to deal with the massive amount of data. If there were, for example, dedicated data stores that provided all of the necessary sequences to every rank – eliminating its dependency on other compute units – there would need to be more cores that would spend the vast majority of their time waiting for messages, which is inefficient.

RayPlatform has a relatively conservative approach when it comes to scheduling. All ranks run the same code, and with the exception of very few moments in the algorithm (when all the work is done, for example), do the same work on different parts of the data. Each rank has an allocated opportunity, once per cycle, to receive a message, and then process it. When more than one message is available, only the first is read. The remaining messages will wait for the next iteration. Each rank has a quantum of work, that is really advancing one of several ongoing tasks, each assigned to one worker of a pre-allocated pool. For example, a quantum of work can consist in extracting a subsequence from a DNA read. The rank then sends messages, and if a task that is part of the rank's work is waiting on another rank's reply, it is suspended and other tasks are chosen instead. The granularity is really small to allow each rank to timely probe incoming messages as computation and communication take place in the same thread on each rank.

A state machine manages the steps of the program. These steps and events are grouped in three types : master modes, slave modes, and message tags. Associated handlers are registered with the RayPlatform API by the application at running time. All steps can be done in parallel, and when all the work for a step is done, and/or additional conditions are met, all ranks

advance to the next state. The advantage of this method is that it is very generic : no prior knowledge regarding data dependencies within a given step of the algorithm is required, or at least these dependencies are delegated from the framework to the application. For example, from a data point-of-view, one first needs to load the sequence reads in memory before building a distributed de Bruijn subgraph. This data dependency is managed by the state machine, and a given rank knows when it's ready. When everyone is ready, the next state is affected. The scheduling scheme in Ray is broadly applicable, provided that tasks can be encoded with granularity.

## 8.11 The true weight of big data

A problem with big data, especially when paired with large compute needs, is that bigger data is harder to move. A single machine can hardly hold all of the data, not even all of the data it needs, at the same time, and depending on data structures and network interconnects, acquiring that data is costly.

There are several ways around this problem. Faster, shared data stores can be used, in the form of large amounts of shared memory or specific, high-performance networked filesystems. They are often expensive, and because they require specific maintenance they are not generally offered as part of cloud services, nor are they available by merely connecting together bits of commodity hardware. For data transport, DNA sequencing instruments can send data in increment in the cloud where they can be processed (examples : Galaxy [97], DNAnexus [15, 228, 179], Illumina BaseSpace [282], GenomeSpace [240]).

The data can be pre-processed and structured, to make retrieval faster, perhaps at the cost of deletion or insertion times. Several solutions oriented towards the big data market exist in terms of providing fast, concurrent databases with good access times [43, 67, 40], and this might be the best option on cloud storage provided there is a value in keeping the structured data after running an algorithm on it.

Finally, locality of similar data can be improved. Depending on the data type, there are algorithms that can do this – it is a solved problem on three-dimensional meshes [150]. This particular approach is very data-dependent, and similarity of the data is often what needs to be computed in the first place. Ray keeps the data stored in the same ranks that are responsible for the computation, distributed uniformly. With fast interconnects or efficient routing algorithms, because the individual vertices in the graph are small, this is not a great problem during execution.

## 8.12 Keeping a large-scale application useful

The proper use of computing time and resources is an area of focus in high-performance computer software research. However, in a world where a machine's time costs less than a dollar an hour, and human wages are climbing globally, it is worthwhile to consider the gain in efficiency from the proper use of human resources. To keep an application useful several levers can be exploited : ease of set-up, accuracy of results, and availability of information are three. We will describe how Ray benefits from its usability and how it gained it.

## 8.13 Human learning : an embarrassingly non-parallel problem

From the standpoint of a software developer, most tasks are ripe for optimization. Data can always be pre-processed, state can always be stored (for example, using checkpoints), and when in doubt, add more cores. This rule does not hold for set-up time, because adding more analysts won't make them understand the software's parameters and chosen file format faster.

Genomics knows many file formats for sequences (fasta, fastq [56], sff [177], color-space [252]), alignments (sam [160], bam [19], psl [138]), variations (vcf [65], gvf [239]), and annotations (gff [254], bed [236], wig). Raw DNA sequences from published research are archived in the Sequence Read Archive [152]. The packaging of data is a problem for big data analysts because in the absence of standards, every software will design their own (often overfitted to their own problem). In the presence of standards, analysts will feel constricted by the data representation and have trouble conceptualizing a way to coerce the results into a format that is usable by other programs.

Ray accepts all kinds of file formats, and outputs in the standard fasta format. Its plugins that perform taxonomic profiling [32] output data in XML. For genome assembly, Ray has only one important parameter : the length of the subsequences ($k$-mers), which will be familiar to anyone experienced with assembly and is easily explained to a biologist (see [18, 58, 85]). Other parameters are adjusted automatically with the data.

## 8.14 Better results or faster results ?

Much is made to improve software performance, especially on pay-per-use resources. More often, users, especially in the scientific field, will demand that the accuracy of the results match that of the more commonplace imperative algorithms they used to run on their desktops [265].

When our team took part in the Assemblathon 2 challenge (http ://assemblathon.org), which aims to identify the best DNA assembly methods for vertebrate genomes, it was found that no assembler managed to obtain consistently best-in-class results on all datasets. However, Ray was ranked first overall for the snake genome dataset [34]

While Ray's primary selling points are its great scalability (see Figure 8.21 and ref. [32]) and portability, not even being fast on larger data can let analysis software off the hook altogether. And we all have to ensure at least state-of-the-art precision. While this can be difficult for particularly interlinked data, the task was facilitated by the abstraction provided by RayPlatform and its state machine.

## 8.15 Being open closes no doors

Ultimately, the best way to make the most of users' time is to listen to them. Ray is open-source software, and in addition the team maintains two mailing lists and takes care of issue tracking. This offers an interface for users to express their problems or propose their ideas, and promotes Ray's (and by extension RayPlatform's) growth towards ever more efficiency by keeping its purpose clear – to provide good data analysis on nearly every machine for every user for problems related to *de novo* genome assembly.

For data analysis and validation, the time cost of having to download the results before looking at them can be prohibitive. We developed Ray Cloud Browser (http ://genome.ulaval.ca :10090/client/ and https ://github.com/sebhtml/Ray-Cloud-Browser), a streaming solution that allows the user to watch the reconstructed genome as an actual graph from any web browser. Ray Cloud Browser is shown in Figure 8.3. This web-based streaming visualizer called Ray Cloud Browser was developed as a companion to the assembler to allow users to walk through assembly paths themselves, just as the algorithm does. Not only is the concept of the assembly problem more clearly expressed when removed from its data-manipulation trappings, the interface allows analysts to publish their results on a server for end users to consume without having to download giant files. Beyond validation, analyses can be performed in a colored version of the graph. Such graphs can be generated by Ray [32], ABySS [276] or by Cortex in the case of multiple samples [126]. This kind of openness – between software providers and analysts, and between analysts and end-users – lead to faster results and will generate faster decisions based on big data.

## 8.16 The road ahead

Improving our Ray genome assembler code base is part of the roadmap. Using Ray Cloud Browser, we are able to visually investigate particular cases in the data where our algorithms

can be improved.

Future work for our framework – RayPlatform – includes load balancing of workers so that processing tasks are migratable between MPI ranks. Previous research papers from the CHARM++ project will assist us in that regard [135, 134].

## 8.17 Conclusions

When dealing with large data and complex systems, what users really want are three attributes : scalability, usability, correctness. To have it all, we have designed RayPlatform, a portable parallel framework on which is based the genome assembler Ray [30, 32]. Other genomic applications can use this framework such as our metagenome assembler and profiler [32]. Our approach does not aim at identifying big data and leveraging bigger systems : rather it deals with many systems, for many data. An interlocked graph is split into tractable chunks of data, and a network of computers into communicable units. With fine granularity, ranks work and communicate as little as possible in the shortest possible amount of time. This allows for scaling both problem and resources from small data, to big data, to bigger data, and beyond, without a paradigm change, allowing the details of the approach to truly fit the problem, and not its size.

We have presented the larger design principles at work in RayPlatform, and the reasoning behind them : getting the most efficiency out of both machine and user, keeping usability and portability at the forefront. As data gets bigger, and analysts scarcer, it's not only compute time that matters.

## 8.18 Acknowledgments

We are grateful to Rick L. Stevens and to Frédéric Raymond for helpful comments. SB is recipient of a Frederick Banting and Charles Best Canada Graduate Scholarship Doctoral Award (200910GSD-226209-172830) from the Canadian Institutes for Health Research (CIHR). JC is the Canada Research Chair in Medical Genomics. MK was supported by Cray Inc. This research was supported in part by the Fonds de recherche du Québec - Nature et technologies (grant 2013-PR-166708 to FL and JC) and by the Discovery Grants Program (Individual, Team and Subatomic Physics Project) from the Natural Sciences and Engineering Research Council of Canada (grant 262067 to FL). Compute time on the Colosse supercomputer at Université Laval was provided by Calcul Québec / Compute Canada (projects nne-790-ab & nne-790-ac). Computations were also carried out on the Mammouth-parallèle II (mp2) supercomputer at Université de Sherbrooke (Calcul Québec / Calcul Canada). The operations of Colosse are funded by the Canada Foundation for Innovation (CFI), the National Science and

Engineering Research Council (NSERC), NanoQuébec, and the Fonds Québécois de Recherche sur la Nature et les Technologies (FQRNT).

## 8.19 Disclosure

The authors declare no competing financial interests exist.

## 8.20 References

[1] Gathering clouds and a sequencing storm. *Nature Biotechnology*, 28(1) :1, January 2010.

[2] Baba Arimilli, Ravi Arimilli, Vicente Chung, Scott Clark, Wolfgang Denzel, Ben Drerup, Torsten Hoefler, Jody Joyner, Jerry Lewis, Jian Li, Nan Ni, and Ram Rajamony. The PERCS High-Performance Interconnect. In *Proceedings of the 2010 18th IEEE Symposium on High Performance Interconnects*, HOTI '10, pages 75–82, Washington, DC, USA, 2010. IEEE Computer Society.

[3] Monya Baker. Next-generation sequencing : adjusting to data overload. *Nature Methods*, 7(7) :495–499, July 2010.

[4] Derek W. Barnett, Erik K. Garrison, Aaron R. Quinlan, Michael P. Strömberg, and Gabor T. Marth. BamTools : a C++ API and toolkit for analyzing and managing BAM files. *Bioinformatics*, 27(12) :1691–1692, June 2011.

[5] Alex Bateman and Matt Wood. Cloud computing. *Bioinformatics*, 25(12) :1475, June 2009.

[6] W. Ben-Ameur. Between fully dynamic routing and robust stable routing. In *Design and Reliable Communication Networks, 2007. DRCN 2007. 6th International Workshop on*, pages 1–6. IEEE, October 2007.

[7] Walid Ben-Ameur and Hervé Kerivin. Routing of Uncertain Traffic Demands. *Optimization and Engineering*, 6(3) :283–313, September 2005.

[8] Walid Ben-Ameur and Mateusz Żotkiewicz. Robust routing and optimal partitioning of a traffic demand polytope. *International Transactions in Operational Research*, 18(3) :307–333, May 2011.

[9] David R. Bentley, Shankar Balasubramanian, Harold P. Swerdlow, Geoffrey P. Smith, John Milton, Clive G. Brown, Kevin P. Hall, Dirk J. Evers, Colin L. Barnes, Helen R.

Bignell, Jonathan M. Boutell, Jason Bryant, Richard J. Carter, R. Keira Cheetham, Anthony J. Cox, Darren J. Ellis, Michael R. Flatbush, Niall A. Gormley, Sean J. Humphray, Leslie J. Irving, Mirian S. Karbelashvili, Scott M. Kirk, Heng Li, Xiaohai Liu, Klaus S. Maisinger, Lisa J. Murray, Bojan Obradovic, Tobias Ost, Michael L. Parkinson, Mark R. Pratt, Isabelle M. J. Rasolonjatovo, Mark T. Reed, Roberto Rigatti, Chiara Rodighiero, Mark T. Ross, Andrea Sabot, Subramanian V. Sankar, Aylwyn Scally, Gary P. Schroth, Mark E. Smith, Vincent P. Smith, Anastassia Spiridou, Peta E. Torrance, Svilen S. Tzonev, Eric H. Vermaas, Klaudia Walter, Xiaolin Wu, Lu Zhang, Mohammed D. Alam, Carole Anastasi, Ify C. Aniebo, David M. D. Bailey, Iain R. Bancarz, Saibal Banerjee, Selena G. Barbour, Primo A. Baybayan, Vincent A. Benoit, Kevin F. Benson, Claire Bevis, Phillip J. Black, Asha Boodhun, Joe S. Brennan, John A. Bridgham, Rob C. Brown, Andrew A. Brown, Dale H. Buermann, Abass A. Bundu, James C. Burrows, Nigel P. Carter, Nestor Castillo, Maria Chiara, Simon Chang, R. Neil Cooley, Natasha R. Crake, Olubunmi O. Dada, Konstantinos D. Diakoumakos, Belen D. Fernandez, David J. Earnshaw, Ugonna C. Egbujor, David W. Elmore, Sergey S. Etchin, Mark R. Ewan, Milan Fedurco, Louise J. Fraser, Karin V. Fuentes Fajardo, W. Scott Furey, David George, Kimberley J. Gietzen, Colin P. Goddard, George S. Golda, Philip A. Granieri, David E. Green, David L. Gustafson, Nancy F. Hansen, Kevin Harnish, Christian D. Haudenschild, Narinder I. Heyer, Matthew M. Hims, Johnny T. Ho, Adrian M. Horgan, Katya Hoschler, Steve Hurwitz, Denis V. Ivanov, Maria Q. Johnson, Terena James, T. A. Huw Jones, Gyoung D. Kang, Tzvetana H. Kerelska, Alan D. Kersey, Irina Khrebtukova, Alex P. Kindwall, Zoya Kingsbury, Paula I. Kokko Gonzales, Anil Kumar, Marc A. Laurent, Cynthia T. Lawley, Sarah E. Lee, Xavier Lee, Arnold K. Liao, Jennifer A. Loch, Mitch Lok, Shujun Luo, Radhika M. Mammen, John W. Martin, Patrick G. McCauley, Paul McNitt, Parul Mehta, Keith W. Moon, Joe W. Mullens, Taksina Newington, Zemin Ning, Bee L. Ng, Sonia M. Novo, Michael J. O'Neill, Mark A. Osborne, Andrew Osnowski, Omead Ostadan, Lambros L. Paraschos, Lea Pickering, Andrew C. Pike, Alger C. Pike, D. Chris Pinkard, Daniel P. Pliskin, Joe Podhasky, Victor J. Quijano, Come Raczy, Vicki H. Rae, Stephen R. Rawlings, Ana C. Rodriguez, Phyllida M. Roe, John Rogers, Maria C. Rogert Bacigalupo, Nikolai Romanov, Anthony Romieu, Rithy K. Roth, Natalie J. Rourke, Silke T. Ruediger, Eli Rusman, Raquel M. Sanches Kuiper, Martin R. Schenker, Josefina M. Seoane, Richard J. Shaw, Mitch K. Shiver, Steven W. Short, Ning L. Sizto, Johannes P. Sluis, Melanie A. Smith, Jean Ernest Sohna Sohna, Eric J. Spence, Kim Stevens, Neil Sutton, Lukasz Szajkowski, Carolyn L. Tregidgo, Gerardo Turcatti, Stephanie Vandevondele, Yuli Verhovsky, Selene M. Virk, Suzanne Wakelin, Gregory C. Walcott, Jingwen Wang, Graham J. Worsley, Juying Yan, Ling Yau, Mike Zuerlein, Jane Rogers, James C. Mullikin, Matthew E. Hurles, Nick J. McCooke, John S. West, Frank L. Oaks, Peter L. Lundberg, David Klenerman, Richard Durbin, and Anthony J. Smith. Ac-

curate whole human genome sequencing using reversible terminator chemistry. *Nature*, 456(7218) :53–59, November 2008.

[10] Inanc Birol, Anthony Raymond, Shaun D. Jackman, Stephen Pleasance, Robin Coope, Greg A. Taylor, Macaire M. Yuen, Christopher I. Keeling, Dana Brand, Benjamin P. Vandervalk, Heather Kirk, Pawan Pandoh, Richard A. Moore, Yongjun Zhao, Andrew J. Mungall, Barry Jaquish, Alvin Yanchuk, Carol Ritland, Brian Boyle, Jean Bousquet, Kermit Ritland, John MacKay, Jörg Bohlmann, and Steven J. M. Jones. Assembling the 20 Gb white spruce (Picea glauca) genome from whole-genome shotgun sequencing data. *Bioinformatics*, 29(12) :1492–1497, June 2013.

[11] Sébastien Boisvert, François Laviolette, and Jacques Corbeil. Ray : simultaneous assembly of reads from a mix of high-throughput sequencing technologies. *Journal of Computational Biology*, 17(11) :1519–1533, November 2010.

[12] Sebastien Boisvert, Frederic Raymond, Elenie Godzaridis, Francois Laviolette, and Jacques Corbeil. Ray Meta : scalable de novo metagenome assembly and profiling. *Genome Biology*, 13(12) :R122+, December 2012.

[13] Keith R. Bradnam, Joseph N. Fass, Anton Alexandrov, Paul Baranay, Michael Bechner, İnanç Birol, Sébastien Boisvert10, Jarrod A. Chapman, Guillaume Chapuis, Rayan Chikhi, Hamidreza Chitsaz, Wen-Chi Chou, Jacques Corbeil, Cristian Del Fabbro, T. Roderick Docking, Richard Durbin, Dent Earl, Scott Emrich, Pavel Fedotov, Nuno A. Fonseca, Ganeshkumar Ganapathy, Richard A. Gibbs, Sante Gnerre, Élénie Godzaridis, Steve Goldstein, Matthias Haimel, Giles Hall, David Haussler, Joseph B. Hiatt, Isaac Y. Ho, Jason Howard, Martin Hunt, Shaun D. Jackman, David B. Jaffe, Erich Jarvis, Huaiyang Jiang, Sergey Kazakov, Paul J. Kersey, Jacob O. Kitzman, James R. Knight, Sergey Koren, Tak-Wah Lam, Dominique Lavenier, François Laviolette, Yingrui Li, Zhenyu Li, Binghang Liu, Yue Liu, Ruibang Luo, Iain MacCallum, Matthew D. MacManes, Nicolas Maillet, Sergey Melnikov, Bruno M. Vieira, Delphine Naquin, Zemin Ning, Thomas D. Otto, Benedict Paten, Octávio S. Paulo, Adam M. Phillippy, Francisco Pina-Martins, Michael Place, Dariusz Przybylski, Xiang Qin, Carson Qu, Filipe J. Ribeiro, Stephen Richards, Daniel S. Rokhsar, J. Graham Ruby, Simone Scalabrin, Michael C. Schatz, David C. Schwartz, Alexey Sergushichev, Ted Sharpe, Timothy I. Shaw, Jay Shendure, Yujian Shi, Jared T. Simpson, Henry Song, Fedor Tsarev, Francesco Vezzi, Riccardo Vicedomini, Jun Wang, Kim C. Worley, Shuangye Yin, Siu-Ming Yiu, Jianying Yuan, Guojie Zhang, Hao Zhang, Shiguo Zhou, and Ian F. Korf1. Assemblathon 2 : evaluating de novo methods of genome assembly in three vertebrate species, January 2013.

[14] Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nilakantan, Arild Skjolsvold, Sam McKelvie, Yikang Xu, Shashwat Srivastav, Jiesheng Wu, Huseyin Simitci, Jaidev Haridas, Cha-

kravarthy Uddaraju, Hemal Khatri, Andrew Edwards, Vaman Bedekar, Shane Mainali, Rafay Abbasi, Arpit Agarwal, Mian Fahim ul Haq, Muhammad Ikram ul Haq, Deepali Bhardwaj, Sowmya Dayanand, Anitha Adusumilli, Marvin McNett, Sriram Sankaran, Kavitha Manivannan, and Leonidas Rigas. Windows Azure Storage : a highly available cloud storage service with strong consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 143–157, New York, NY, USA, 2011. ACM.

[15] J. Gregory Caporaso, Justin Kuczynski, Jesse Stombaugh, Kyle Bittinger, Frederic D. Bushman, Elizabeth K. Costello, Noah Fierer, Antonio G. Pena, Julia K. Goodrich, Jeffrey I. Gordon, Gavin A. Huttley, Scott T. Kelley, Dan Knights, Jeremy E. Koenig, Ruth E. Ley, Catherine A. Lozupone, Daniel McDonald, Brian D. Muegge, Meg Pirrung, Jens Reeder, Joel R. Sevinsky, Peter J. Turnbaugh, William A. Walters, Jeremy Widmann, Tanya Yatsunenko, Jesse Zaneveld, and Rob Knight. QIIME allows analysis of high-throughput community sequencing data. *Nature Methods*, 7(5) :335–336, May 2010.

[16] Rick Cattell. Scalable SQL and NoSQL data stores. *SIGMOD Rec.*, 39(4) :12–27, May 2011.

[17] Dong Chen, Noel A. Eisley, Philip Heidelberger, Robert M. Senger, Yutaka Sugawara, Sameer Kumar, Valentina Salapura, David L. Satterfield, Burkhard S. Burow, and Jeffrey J. Parker. The IBM Blue Gene/Q interconnection network and message unit. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, New York, NY, USA, 2011. ACM.

[18] Rayan Chikhi and Paul Medvedev. Informed and automated k-mer size selection for genome assembly. *Bioinformatics*, pages btt310+, June 2013.

[19] Peter J. A. Cock, Christopher J. Fields, Naohisa Goto, Michael L. Heuer, and Peter M. Rice. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, 38(6) :1767–1771, April 2010.

[20] Phillip E. C. Compeau, Pavel A. Pevzner, and Glenn Tesler. How to apply de Bruijn graphs to genome assembly. *Nature Biotechnology*, 29(11) :987–991, November 2011.

[21] The Human Microbiome Project Consortium. Structure, function and diversity of the healthy human microbiome. *Nature*, 486(7402) :207–214, June 2012.

[22] Elizabeth K. Costello, Christian L. Lauber, Micah Hamady, Noah Fierer, Jeffrey I. Gordon, and Rob Knight. Bacterial Community Variation in Human Body Habitats Across Space and Time. *Science*, 326(5960) :1694–1697, December 2009.

[23] Petr Danecek, Adam Auton, Goncalo Abecasis, Cornelis A. Albers, Eric Banks, Mark A. DePristo, Robert Handsaker, Gerton Lunter, Gabor Marth, Stephen T. Sherry, Gilean McVean, Richard Durbin, and 1000 Genomes Project Analysis Group. The Variant Call Format and VCFtools. *Bioinformatics*, 27(15) :btr330–2158, June 2011.

[24] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo : amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41(6) :205–220, October 2007.

[25] Joel T. Dudley and Atul J. Butte. In silico research in the era of cloud computing. *Nature Biotechnology*, 28(11) :1181–1185, November 2010.

[26] Greg Faanes, Abdulla Bataineh, Duncan Roweth, Tom Court, Edwin Froese, Bob Alverson, Tim Johnson, Joe Kopnick, Mike Higgins, and James Reinhard. Cray cascade : a scalable HPC system based on a Dragonfly network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[27] Paul Flicek and Ewan Birney. Sense from sequence reads : methods for alignment and assembly. *Nature Methods*, 6(11 Suppl) :S6–S12, November 2009.

[28] Message Passing Interface Forum. MPI : A Message-Passing Interface Standard Version 3.0, 2012.

[29] J. K. Gallant. The complexity of the overlap method for sequencing biopolymers. *J Theor Biol*, 101(1) :1–17, March 1983.

[30] Jeremy Goecks, Anton Nekrutenko, James Taylor, and The Galaxy Team. Galaxy : a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8) :R86+, August 2010.

[31] Christopher S. Henry, Matthew DeJongh, Aaron A. Best, Paul M. Frybarger, Ben Linsay, and Rick L. Stevens. High-throughput generation, optimization and analysis of genome-scale metabolic models. *Nat Biotech*, 28(9) :977–982, September 2010.

[32] G. Hutchinson. Evaluation of polymer sequence fragment data using graph theory. *Bulletin of Mathematical Biophysics*, 31(3) :541–562, 1969.

[33] R. M. Idury and M. S. Waterman. A new algorithm for DNA sequence assembly. *Journal of Computational Biology*, 2(2) :291–306, 1995.

[34] Zamin Iqbal, Mario Caccamo, Isaac Turner, Paul Flicek, and Gil McVean. De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nature Genetics*, 44(2) :226–232, February 2012.

[35] Shaun Jackman and Inanc Birol. Assembling genomes using short-read sequencing technology. *Genome Biology*, 11(1) :202+, 2010.

[36] Benjamin G. Jackson, Patrick S. Schnable, and Srinivas Aluru. Parallel short sequence assembly of transcriptomes. *BMC bioinformatics*, 10 Suppl 1(Suppl 1) :S14+, 2009.

[37] BenjaminG Jackson, PatrickS Schnable, and Srinivas Aluru. Assembly of Large Genomes from Paired Short Reads. In Sanguthevar Rajasekaran, editor, *Bioinformatics and Computational Biology*, volume 5462 of *Lecture Notes in Computer Science*, pages 30–43. Springer Berlin Heidelberg, 2009.

[38] Laxmikant V. Kale and Sanjeev Krishnan. CHARM++ : a portable concurrent object oriented system based on C++. In *Proceedings of the eighth annual conference on Object-oriented programming systems, languages, and applications*, OOPSLA '93, pages 91–108, New York, NY, USA, 1993. ACM.

[39] Laxmikant V. Kale and Gengbin Zheng. Charm++ and AMPI : Adaptive Runtime Strategies via Migratable Objects. pages 265–282, 2009.

[40] RichardM Karp. Reducibility Among Combinatorial Problems. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 219–241. Springer Berlin Heidelberg, 2010.

[41] W. James Kent. BLAT–the BLAST-like alignment tool. *Genome Res*, 12(4) :656–664, April 2002.

[42] Donald E. Knuth. *Art of Computer Programming, Volume 3 : Sorting and Searching (2nd Edition)*. Addison-Wesley Professional, 2 edition, May 1998.

[43] Konstantinos Krampis, Tim Booth, Brad Chapman, Bela Tiwari, Mesude Bicak, Dawn Field, and Karen E. Nelson. Cloud BioLinux : pre-configured and on-demand bioinformatics computing for the genomics community. *BMC bioinformatics*, 13(1) :42+, March 2012.

[44] J. K. Lawder and P. J. H. King. Querying multi-dimensional data indexed using the Hilbert space-filling curve. *SIGMOD Rec.*, 30(1) :19–24, March 2001.

[45] Rasko Leinonen, Hideaki Sugawara, and Martin Shumway. The Sequence Read Archive. *Nucleic Acids Research*, 39(suppl 1) :D19–D21, January 2011.

[46] Ruth E. Ley, Catherine A. Lozupone, Micah Hamady, Rob Knight, and Jeffrey I. Gordon. Worlds within worlds : evolution of the vertebrate gut microbiota. *Nature Reviews Microbiology*, 6(10) :776–788, October 2008.

[47] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, and 1000 Genome Project Data Processing Subgroup. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16) :2078–2079, August 2009.

[48] Chris Loken, Daniel Gruner, Leslie Groer, Richard Peltier, Neil Bunn, Michael Craig, Teresa Henriques, Jillian Dempsey, Ching-Hsing Yu, Joseph Chen, L. Jonathan Dursi, Jason Chong, Scott Northrup, Jaime Pinto, Neil Knecht, and Ramses Van Zon. SciNet : Lessons Learned from Building a Power-efficient Top-20 System and Data Centre. *Journal of Physics : Conference Series*, 256(1) :012026+, December 2010.

[49] Catherine Lozupone, Micah Hamady, and Rob Knight. UniFrac–an online tool for comparing microbial community diversity in a phylogenetic context. *BMC bioinformatics*, 7(1) :371+, August 2006.

[50] Catherine Lozupone and Rob Knight. UniFrac : a New Phylogenetic Method for Comparing Microbial Communities. *Applied and Environmental Microbiology*, 71(12) :8228–8235, December 2005.

[51] Catherine A. Lozupone and Rob Knight. Global patterns in bacterial diversity. *Proceedings of the National Academy of Sciences*, 104(27) :11436–11440, July 2007.

[52] Elaine Mardis. The $1,000 genome, the $100,000 analysis ? *Genome Medicine*, 2(11) :84+, 2010.

[53] Elaine R. Mardis. New strategies and emerging technologies for massively parallel sequencing : applications in medical research. *Genome medicine*, 1(4), April 2009.

[54] Marcel Margulies, Michael Egholm, William E. Altman, Said Attiya, Joel S. Bader, Lisa A. Bemben, Jan Berka, Michael S. Braverman, Yi-Ju Chen, Zhoutao Chen, Scott B. Dewell, Lei Du, Joseph M. Fierro, Xavier V. Gomes, Brian C. Godwin, Wen He, Scott Helgesen, Chun H. Ho, Gerard P. Irzyk, Szilveszter C. Jando, Maria L. I. Alenquer, Thomas P. Jarvie, Kshama B. Jirage, Jong-Bum Kim, James R. Knight, Janna R. Lanza, John H. Leamon, Steven M. Lefkowitz, Ming Lei, Jing Li, Kenton L. Lohman, Hong Lu, Vinod B. Makhijani, Keith E. Mcdade, Michael P. Mckenna, Eugene W. Myers, Elizabeth Nickerson, John R. Nobile, Ramona Plant, Bernard P. Puc, Michael T. Ronan, George T. Roth, Gary J. Sarkis, Jan F. Simons, John W. Simpson, Maithreyan Srinivasan, Karrie R. Tartaro, Alexander Tomasz, Kari A. Vogt, Greg A. Volkmer, Shally H. Wang,

Yong Wang, Michael P. Weiner, Pengguang Yu, Richard F. Begley, and Jonathan M. Rothberg. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057) :376–380, July 2005.

[55] Vivien Marx. Biology : The big challenges of big data. *Nature*, 498(7453) :255–260, June 2013.

[56] Vivien Marx. Genomics in the clouds. *Nature Methods*, 10(10) :941–945, September 2013.

[57] John D. McPherson. Next-generation gap. *Nature Methods*, 6(11 Suppl) :S2–S5, November 2009.

[58] Jason R. Miller, Sergey Koren, and Granger Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6) :315–327, June 2010.

[59] Eugene W. Myers. The fragment assembly string graph. *Bioinformatics*, 21 Suppl 2 :ii79–ii85, September 2005.

[60] Eugene W. Myers, Granger G. Sutton, Hamilton O. Smith, Mark D. Adams, and J. Craig Venter. On the sequencing and assembly of the human genome. *Proc Natl Acad Sci U S A*, 99(7) :4145–4146, April 2002.

[61] Shota Nakamura, Norihiro Maeda, Ionut Mihai M. Miron, Myonsun Yoh, Kaori Izutsu, Chidoh Kataoka, Takeshi Honda, Teruo Yasunaga, Takaaki Nakaya, Jun Kawai, Yoshihide Hayashizaki, Toshihiro Horii, and Tetsuya Iida. Metagenomic diagnosis of bacterial infections. *Emerging infectious diseases*, 14(11) :1784–1786, November 2008.

[62] P. A. Pevzner. 1-Tuple DNA sequencing : computer analysis. *J Biomol Struct Dyn*, 7(1) :63–73, August 1989.

[63] P. A. Pevzner, H. Tang, and M. S. Waterman. An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17) :9748–9753, 2001.

[64] C. D. Pham. Comparison of Message Aggregation Strategies for Parallel Simulations on a High Performance Cluster. In *IN PROCEEDINGS OF THE 8TH INTERNATIONAL SYMPOSIUM ON MODELING, ANALYSIS AND SIMULATION OF COMPUTER AND TELECOMMUNICATION SYSTEMS, AUGUST-SEPTEMBER*, 2000.

[65] Andrew Pollack. DNA sequencing caught in deluge of data. *New York Times*, 1, 2011.

[66] Mihai Pop. Genome assembly reborn : recent computational challenges. *Brief Bioinform*, 10(4) :354–366, July 2009.

[67] Oliver G. Pybus and Andrew Rambaut. Evolutionary analysis of the dynamics of viral infectious disease. *Nat Rev Genet*, 10(8) :540–550, August 2009.

[68] Xiaohong Qiu, Jaliya Ekanayake, Scott Beason, Thilina Gunarathne, Geoffrey Fox, Roger Barga, and Dennis Gannon. Cloud technologies for bioinformatics applications. In *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers*, MTAGS '09, New York, NY, USA, 2009. ACM.

[69] Aaron R. Quinlan and Ira M. Hall. BEDTools : a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6) :841–842, March 2010.

[70] Martin Reese, Barry Moore, Colin Batchelor, Fidel Salas, Fiona Cunningham, Gabor Marth, Lincoln Stein, Paul Flicek, Mark Yandell, and Karen Eilbeck. A standard variation file format for human genome sequences. *Genome Biology*, 11(8) :R88+, August 2010.

[71] Michael Reich, John Liefeld, Helga Thorvaldsdottir, Marco Ocana, Eliot Polk, D. K. Jang, and Jill Mesirov. GenomeSpace : An environment for frictionless bioinformatics. In *Proceedings of the 103rd Annual Meeting of the American Association for Cancer Research*, volume 72, pages 3966+. American Association for Cancer Research, April 2012.

[72] Jonathan M. Rothberg and John H. Leamon. The development and impact of 454 sequencing. *Nat Biotechnol*, 26(10) :1117–1124, October 2008.

[73] Stephen M. Rumble, Phil Lacroute, Adrian V. Dalca, Marc Fiume, Arend Sidow, and Michael Brudno. SHRiMP : Accurate Mapping of Short Color-space Reads. *PLoS Comput Biol*, 5(5) :e1000386+, May 2009.

[74] Kim Rutherford, Julian Parkhill, James Crook, Terry Horsnell, Peter Rice, Marie-Adèle Rajandream, and Bart Barrell. Artemis : sequence visualization and annotation. *Bioinformatics*, 16(10) :944–945, October 2000.

[75] Eric E. Schadt, Michael D. Linderman, Jon Sorenson, Lawrence Lee, and Garry P. Nolan. Computational solutions to large-scale data management and analysis. *Nature Reviews Genetics*, 11(9) :647–657, September 2010.

[76] Michael C. Schatz, Ben Langmead, and Steven L. Salzberg. Cloud computing and the DNA data race. *Nature Biotechnology*, 28(7) :691–693, July 2010.

[77] Steven L. Scott and Et Al. The Cray T3E Network : Adaptive Routing in a High Performance 3D Torus. 1996.

[78] Jared T. Simpson, Kim Wong, Shaun D. Jackman, Jacqueline E. Schein, Steven J. M. Jones, and Inanç Birol. ABySS : A parallel assembler for short read sequence data. *Genome Research*, 19(6) :1117–1123, 2009.

[79] Chris S. Smillie, Mark B. Smith, Jonathan Friedman, Otto X. Cordero, Lawrence A. David, and Eric J. Alm. Ecology drives a global network of gene exchange connecting the human microbiome. *Nature*, 480(7376) :241–244, December 2011.

[80] Lincoln Stein. Creating a bioinformatics nation. *Nature*, 417(6885) :119–120, May 2002.

[81] Lincoln D. Stein. The case for cloud computing in genome informatics. *Genome Biology*, 11(5) :207+, May 2010.

[82] Jordan Stockton. BaseSpace Roadmap. In *Plant and Animal Genome XXI Conference*. Plant and Animal Genome, 2013.

[83] Dongying Wu, Philip Hugenholtz, Konstantinos Mavromatis, Rudiger Pukall, Eileen Dalin, Natalia N. Ivanova, Victor Kunin, Lynne Goodwin, Martin Wu, Brian J. Tindall, Sean D. Hooper, Amrita Pati, Athanasios Lykidis, Stefan Spring, Iain J. Anderson, Patrik D/'haeseleer, Adam Zemla, Mitchell Singer, Alla Lapidus, Matt Nolan, Alex Copeland, Cliff Han, Feng Chen, Jan-Fang Cheng, Susan Lucas, Cheryl Kerfeld, Elke Lang, Sabine Gronow, Patrick Chain, David Bruce, Edward M. Rubin, Nikos C. Kyrpides, Hans-Peter Klenk, and Jonathan A. Eisen. A phylogeny-driven genomic encyclopaedia of Bacteria and Archaea. *Nature*, 462(7276) :1056–1060, December 2009.

## 8.21 Figures

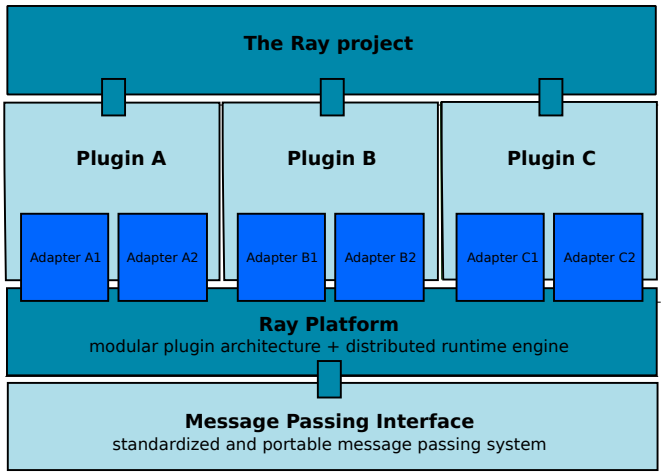


FIGURE 8.1: Relationship between Ray and RayPlatform.

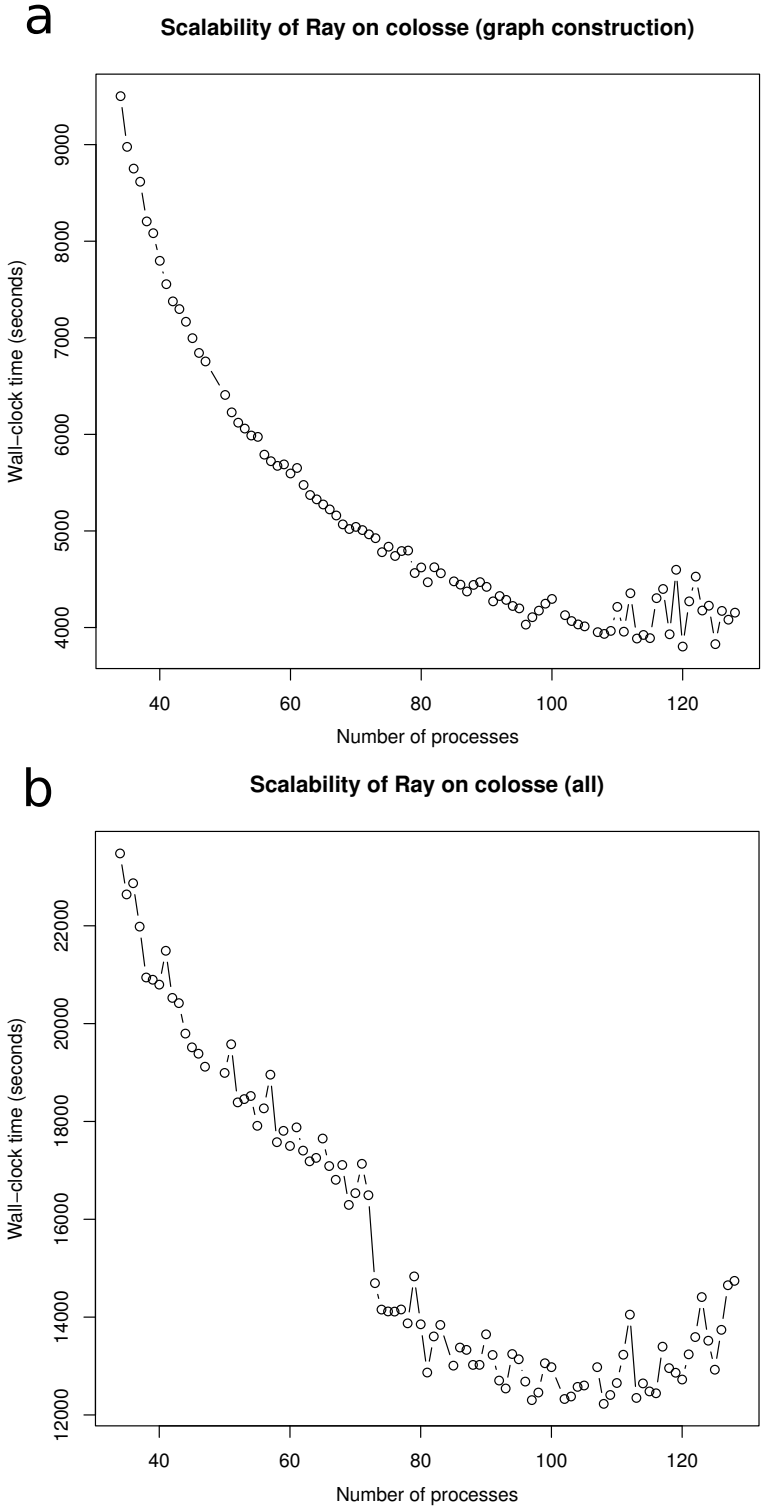


FIGURE 8.2: Scalability of Ray.

The scalability was measured on the sample SRS011098 from the Human Microbiome Project [59]. Using the same input data, different times were obtained by varying the number of processes. Panel a shows the time required to build the genome graph for various numbers of processes. In this panel, the time asymptote is at 4,000 seconds because some parts of the computation are not parallel (Amdahl's law). In panel b, the time asymptote is reached at a lower number of processes because the time includes all the compute steps of an assembly with Ray.

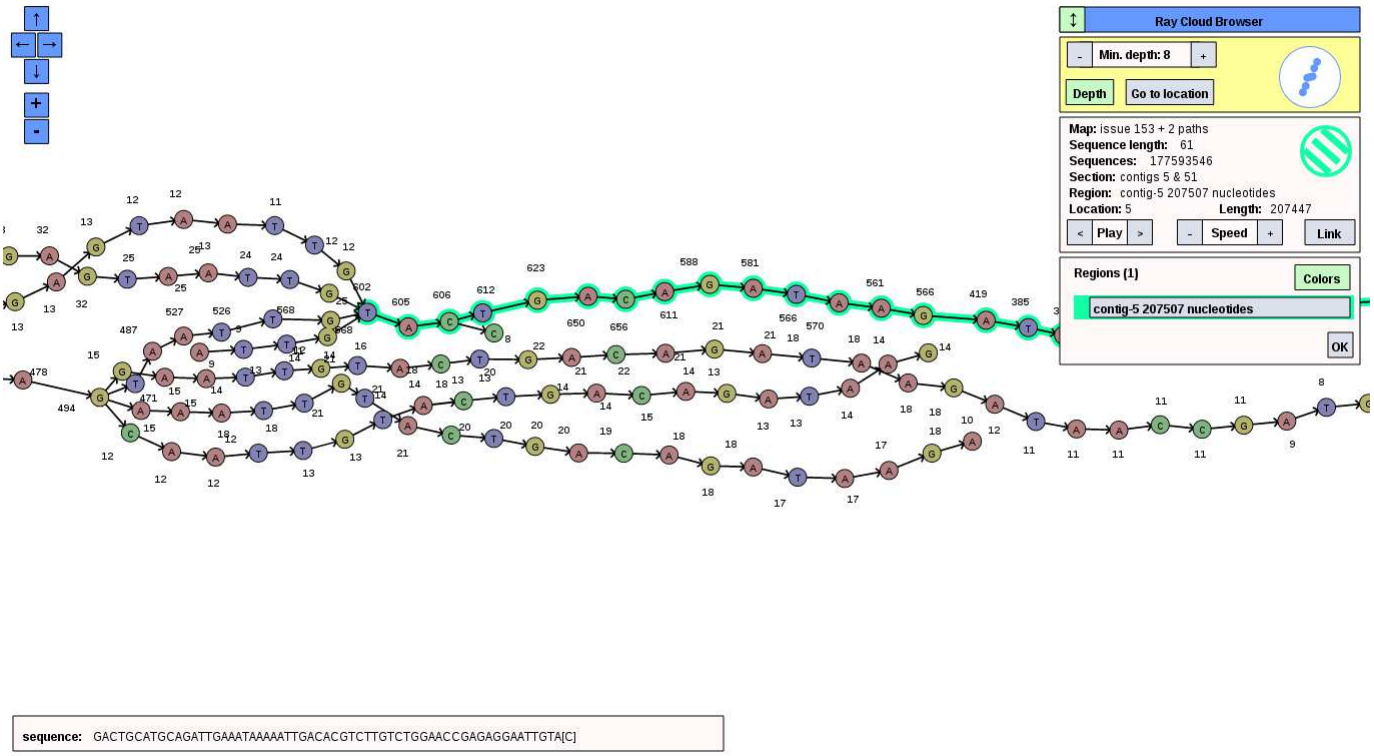


FIGURE 8.3: A screenshot of Ray Cloud Browser.

The green path is a sequence assembled by Ray from reads. Numbers near vertices are sequencing depths. Vertices with low sequencing depths are sequencing errors.

# Chapitre 9

# Collaborations

Ce chapitre contient les articles auxquels j'ai collaboré. Trois (3) des articles ont été faits avant ma maîtrise et mon doctorat, mais sont tout de même pertinents à mentionner. Ce sont des collaborations en génomique. Mes publications sont disponibles à l'adresse http ://bois-vert.info/publications.html .

### 9.0.1 Collaboration 1 : Genome-wide gene expression profiling analysis of Leishmania major and Leishmania infantum developmental stages reveals substantial differences between the two species

**Article** Nous avons publié cet article avant mon doctorat dans la revue *BMC Genomics*. Dans cet article, nous avons montré que le programme d'expression génique de *Leishmania* est différent entre les formes amastigote et promastigote [244].

**Contribution** Dans ce travail, j'ai implémenté un logiciel de gestion d'information de laboratoire dans le langage Ruby on Rails.

### 9.0.2 Collaboration 2 : Modulation of gene expression in drug resistant Leishmania is associated with gene amplification, gene deletion and chromosome aneuploidy

**Article** L'article de Ubeda *et al.* a été publié dans *Genome Biology* en 2008 [289]. Dans ce travail, nous avons démontré que la modulation de l'expression des gènes chez *Leishmania* est associée à l'amplification de gènes, à la délétion de gènes et à l'aneuploïdie. Cet article n'a pas été publié lors de mon doctorat, mais est tout de même pertinent puisque notre plus récente

publication dans *Molecular Microbiology* traite d'un sujet similaire [193].

**Contribution** Dans ce travail, nous avons utilisé le même logiciel de gestion d'information de laboratoire décrit dans la section précédente.

### 9.0.3 Collaboration 3 : Comparison of automated microarray detection with real-time PCR assays for detection of respiratory viruses in specimens obtained from children

**Article** Nous avons publié cet article avant mon doctorat. Nous avons comparé des méthodes de détection pour une sélection de virus respiratoires [238]. Cet article est paru dans le *Journal of Clinical Microbiology*.

**Contribution** Ma contribution à ce projet a été de programmer, déployer, et maintenir un système de gestion d'information de laboratoire. Ce système était un projet différent du projet de système de gestion d'information de laboratoire pour *Leishmania*.

### 9.0.4 Collaboration 4 : Genome sequencing of the lizard parasite Leishmania tarentolae reveals loss of genes associated to the intracellular stage of human pathogenic species

**Article** Nous avons séquencé, assemblé et annoté le génome de *Leishmania tarentolae*. La publication est parue dans le journal *Nucleic Acids Research* [238].

Les séquences ont été obtenues avec la technologie de séquençage 454 [177]. Je suis second auteur sur ce manuscrit.

**Contribution** Ma contribution a été d'assembler le génome à l'aide d'un assembleur.

### 9.0.5 Collaboration 5 : Endonucleases : tools to correct the dystrophin gene

**Article** Nous avons utilisé les méga-nucléases pour éditer de l'ADN génomique dans des cellules afin de corriger des mutations [251].

Dans ce travail, j'ai aidé l'équipe de Jacques P. Tremblay à mesurer l'efficacité des méga-nucléases à réparer des régions géniques impliquées dans la dystrophie musculaire [251]. L'article a été publié dans le *Journal of Gene Medicine*. Cet article est la suite logique d'un autre article publié par le groupe de Jacques P. Tremblay auquel je n'ai pas participé [50]. Depuis la

publication de cette recherche, une nouvelle méthode moins dispendieuse, plus rapide et plus efficace a été développée par l'équipe de George Church [174]. Cette méthode utilise l'enzyme Cas9 du système CRISPR (Cas veut dire *CRISPR-associated system*) ainsi qu'une séquence guide constituée d'ARN.

**Contribution** Ma contribution à ce projet est d'avoir écrit un outil en C++ pour compter chaque type de variations génétiques présentes dans des gènes précis afin d'évaluer l'efficacité des méga-nucléases à réparer des gènes. Les outils standards (comme l'aligneur bwa) ne fonctionnaient pas car les délétions dans l'ADN étaient trop longues.

### 9.0.6 Collaboration 6 : Multiple Mutations in Heterogeneous Miltefosine-Resistant Leishmania major Population as Determined by Whole Genome Sequencing

**Article** En collaboration avec le groupe de recherche de Marc Ouellette, nous avons publié un travail décrivant l'impact de mutations dans le génome de *Leishmania major* [57]. Cet article a été publié dans le journal *PLoS Neglected Tropical Diseases*, une revue relativement prestigieuse dans le domaine.

**Contribution** J'ai fait toutes les analyses bio-informatiques avec les logiciels bwa [159], samtools [160] et vcfutils [65]. J'ai géré les données, et présenté les résultats aux autres membres du projet. J'ai aussi rédigé toute la méthode bio-informatique dans le manuscrit.

### 9.0.7 Collaboration 7 : Telomeric gene deletion and intrachromosomal amplification in antimony resistant Leishmania

**Article** En collaboration avec le groupe de Marc Ouellette, nous avons analysé le nombre de copies de différentes parties de l'architecture génomique du parasite *Leishmania major* [193]. Cet article est paru dans le journal *Molecular Microbiology*.

**Contribution** J'ai fait toutes les analyses bio-informatiques (alignements, nombre de copies, variations, normalisation du signal, revue de la littérature).

### 9.0.8 Collaboration 8 : Assemblathon 2 : evaluating de novo methods of genome assembly in three vertebrate species

**Article** L'assemblathon est un marathon d'assembleurs de génomes. L'article a été mis initialement dans arXiv [34] et est maintenant disponible dans la revue *GigaScience* [33]. Les

participants au projet Assemblathon 1 ont publié un article sur les résultats de l'Assemblathon 1, le prédécesseur de l'Assemblathon 2 [78]. Le premier article de comparaison de méthodes d'assemblage *de novo* a été publié en 1994 [191]. D'autres articles comparant les assembleurs *de novo* ont été publiés aussi [295]. D'autres travaux récents sur l'analyse de la qualité des assemblages incluent GAGE [259], GAGE-B [173], QUAST [104], et REAPR [120]. Ces travaux ont permis de systématiquement évaluer les erreurs d'assemblage [260].

**Contribution** J'étais le responsable à l'Université Laval pour l'Assemblathon 2. Nous avons soumis des assemblages faits avec Ray. J'ai téléchargé les données, fait les assemblages, modifié le code source de Ray, participé aux discussions avec les autres participants (environ 90 participants et participantes).

# Conclusion

## Problème initial

L'objectif de recherche initiale était de concevoir des outils utiles, et meilleurs que ceux existant pour adresser plusieurs questions biologiques. Cet objectif a été atteint.

Dorénavant, il est possible de faire des assemblages *de novo* de génomes et de méta-génomes en parallèle en utilisant le logiciel Ray.

## Résumé du travail

Dans notre article *Retrovirology* (cité 36 fois), nous avons avancé la frontière pour la prédiction de corecepteur du virus de l'immunodéficience humaine (voir la référence [31] et le chapitre 5). Ce type d'approches est utilisé en clinique. L'approche utilisée principalement en clinique est geno2pheno [153], malgré que l'exactitude de cette méthode (91.56%, voir [261]) soit inférieure à l'exactitude de notre méthode (94.80%, voir [31]).

Notre article paru dans *Journal of Computational Biology* a proposé une meilleure méthode pour assembler des génomes et est cité 73 fois [30] (voir le chapitre 6). Nous sommes en train de tester cette méthode sur Titan, le deuxième plus puissant superordinateur au monde.

Notre article *Genome Biology* est cité 11 fois [32] (voir le chapitre 7) et a avancé la frontière pour l'assemblage *de novo* de méta-génomes. Grâce à ce travail, j'ai gagné le prix d'étudiant-chercheur étoile dans la catégorie santé des Fonds de recherche du Québec et j'ai été invité à présenter à l'Université de Liverpool.

Nous avons partagé nos opinions sur la génomique à haute vitesse dans notre papier accepté dans la revue *Big Data* (voir le chapitre 8).

De plus, nous avons participé à l'Assemblathon 2 [33] – une compétition d'assemblage de génomes. Il y avait des données pour 3 types d'animaux : un serpent, un oiseau et un poisson. Une des conclusions de cette compétition est que pour les données de serpent : *The Ray assem-*

*bly was ranked 1st overall, and also ranked 1st for all individual measures except multiplicity (where it still had a better than average performance).*

# Perspectives

## Parallélisme

Après mon doctorat, je vais définitivement continuer mon aventure dans la bio-informatique dans le sentier du passage de messages puisque c'est une avenue très peu fréquentée au potentiel extraordinaire. Les logiciels distribués sont aux organismes multi-cellulaires ce que sont les logiciels séquentiels aux organismes uni-cellulaires.

## Visualisation

Je crois aussi qu'il faut davantage de visualisation pour aller rejoindre les gens de tous les domaines et de toutes les frontières de la science et de la société. L'assemblage *de novo* est difficile de part la présence de régions répétées. Les travaux de 2004 Pevzner sont dans ce sens un bon point de départ [223]. La visualisation de variations complexes aidera au développement de meilleurs outils pour cataloguer la diversité et les différences. Je vais poursuivre le projet Ray Cloud Browser afin de visualiser les problèmes importants. Ceci pourrait déboucher à des algorithmes de reconnaissance de structures biologiques (parties de graphes) qui pourraient assister les biologistes dans la compréhension de phénomènes complexes.

## Intégration

L'intégration de beaucoup de données dans une seule analyse pourra aussi révéler des informations utiles qui demeurent autrement cachées dans les données. Par exemple, l'ajout de différent niveaux entre le génotype et le phénotype (transcrits, métabolites, ou autres objets) directement dans le graphe de de Bruijn pourrait s'avérer utile.

## Comparaison

Par exemple, l'analyse simultanée de plusieurs échantillons dans un seul et même graphe de de Bruijn colorié facilitera les analyses inter-échantillons. Pour des gros jeux de données, comme une centaine d'échantillons, il est nécessaire d'exploiter une architecture distribuée. Une telle architecture a été proposée dans cette thèse avec le cadriciel RayPlatform et démontrée au travers du logiciel Ray pour la génomique.

Récemment, j'ai travaillé sur Ray Surveyor pour comparer les échantillons de l'éclosion de légionellose de Québec. Avec cet outil – qui est encore en développement – il est possible de comparer rapidement et sans génome de référence des échantillons.

## La prochaine génération

La prochaine évolution d'une telle architecture pour la génomique passe cependant par le formalisme des acteurs [112, 5]. J'ai commencé à travailler sur la mise en place du modèle par acteurs dans la plate-forme RayPlatform.

Les logiciels futurs pour la génomique devront donc être massivement parallèles afin de suivre la progression technologique des séquençeurs d'ADN.

# Bibliographie


[1] Gathering clouds and a sequencing storm. *Nature Biotechnology*, 28(1) :1, January 2010.

[2] Enis Afgan, Dannon Baker, Nate Coraor, Brad Chapman, Anton Nekrutenko, and James Taylor. Galaxy CloudMan : delivering cloud compute clusters. *BMC Bioinformatics*, 11(Suppl 12) :S4+, 2010.

[3] Enis Afgan, Dannon Baker, Nate Coraor, Hiroki Goto, Ian M. Paul, Kateryna D. Makova, Anton Nekrutenko, and James Taylor. Harnessing cloud computing with Galaxy Cloud. *Nature Biotechnology*, 29(11) :972–974, November 2011.

[4] Enis Afgan, Brad Chapman, and James Taylor. CloudMan as a platform for tool, data, and analysis distribution. *BMC bioinformatics*, 13(1) :315+, November 2012.

[5] Gul Agha. *Actors : a model of concurrent computation in distributed systems*. MIT Press, Cambridge, MA, USA, 1986.

[6] Gul Agha. An overview of actor languages. In *Proceedings of the 1986 SIGPLAN workshop on Object-oriented programming*, OOPWORK '86, pages 58–67, New York, NY, USA, 1986. ACM.

[7] K. Scheibye Alsing, S. Hoffmann, A. Frankel, P. Jensen, P. F. Stadler, Y. Mang, N. Tommerup, M. J. Gilchrist, A. B. Nygård, S. Cirera, C. B. Jørgensen, M. Fredholm, and J. Gorodkin. Sequence assembly. *Comput Biol Chem*, 33(2) :121–136, April 2009.

[8] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST : a new generation of protein database search programs. *Nucleic Acids Res.*, 25 :3389–3402, September 1997.

[9] Sasha K. Ames, David A. Hysom, Shea N. Gardner, G. Scott Lloyd, Maya B. Gokhale, and Jonathan E. Allen. Scalable metagenomic taxonomy classification using a reference genome database. *Bioinformatics (Oxford, England)*, 29(18) :2253–2260, September 2013.

[10] Baba Arimilli, Ravi Arimilli, Vicente Chung, Scott Clark, Wolfgang Denzel, Ben Drerup, Torsten Hoefler, Jody Joyner, Jerry Lewis, Jian Li, Nan Ni, and Ram Rajamony. The PERCS High-Performance Interconnect. In *Proceedings of the 2010 18th IEEE Symposium on High Performance Interconnects*, HOTI '10, pages 75–82, Washington, DC, USA, 2010. IEEE Computer Society.

[11] Manimozhiyan Arumugam, Jeroen Raes, Eric Pelletier, Denis Le Paslier, Takuji Yamada, Daniel R. Mende, Gabriel R. Fernandes, Julien Tap, Thomas Bruls, Jean-Michel Batto, Marcelo Bertalan, Natalia Borruel, Francesc Casellas, Leyden Fernandez, Laurent Gautier, Torben Hansen, Masahira Hattori, Tetsuya Hayashi, Michiel Kleerebezem, Ken Kurokawa, Marion Leclerc, Florence Levenez, Chaysavanh Manichanh, H. Bjorn Nielsen, Trine Nielsen, Nicolas Pons, Julie Poulain, Junjie Qin, Thomas Sicheritz-Ponten, Sebastian Tims, David Torrents, Edgardo Ugarte, Erwin G. Zoetendal, Jun Wang, Francisco Guarner, Oluf Pedersen, Willem M. de Vos, Soren Brunak, Joel Dore, Jean Weissenbach, S. Dusko Ehrlich, and Peer Bork. Enterotypes of the human gut microbiome. *Nature*, 473(7346) :174–180, May 2011.

[12] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene ontology : tool for the unification of biology. The Gene Ontology Consortium. *Nature Genetics*, 25(1) :25–29, May 2000.

[13] Jean M. Aury, Corinne Cruaud, Valérie Barbe, Odile Rogier, Sophie Mangenot, Gaelle Samson, Julie Poulain, Véronique Anthouard, Claude Scarpelli, François Artiguenave, and Patrick Wincker. High quality draft sequences for prokaryotic genomes using a mix of new sequencing technologies. *BMC Genomics*, 9 :603, 2008.

[14] Ramy K. Aziz, Scott Devoid, Terrence Disz, Robert A. Edwards, Christopher S. Henry, Gary J. Olsen, Robert Olson, Ross Overbeek, Bruce Parrello, Gordon D. Pusch, Rick L. Stevens, Veronika Vonstein, and Fangfang Xia. SEED Servers : High-Performance Access to the SEED Genomes, Annotations, and Metabolic Models. *PLoS ONE*, 7(10) :e48053+, October 2012.

[15] Monya Baker. Next-generation sequencing : adjusting to data overload. *Nature Methods*, 7(7) :495–499, July 2010.

[16] Satish Balay, William D. Gropp, Lois C. McInnes, and Barry F. Smith. Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[17] Anton Bankevich, Sergey Nurk, Dmitry Antipov, Alexey A. Gurevich, Mikhail Dvorkin, Alexander S. Kulikov, Valery M. Lesin, Sergey I. Nikolenko, Son Pham, Andrey D. Prjibelski, Alexey V. Pyshkin, Alexander V. Sirotkin, Nikolay Vyahhi, Glenn Tesler, Max A. Alekseyev, and Pavel A. Pevzner. SPAdes : a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of computational biology : a journal of computational molecular cell biology*, 19(5) :455–477, May 2012.

[18] Albert-Laszlo Barabasi and Zoltan N. Oltvai. Network biology : understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2) :101–113, February 2004.

[19] Derek Barnett, Erik Garrison, Aaron Quinlan, Michael Strömberg, and Gabor Marth. BamTools : a C++ API and toolkit for analyzing and managing BAM files. *Bioinformatics*, 27(12) :btr174–1692, April 2011.

[20] Alex Bateman and Matt Wood. Cloud computing. *Bioinformatics*, 25(12) :1475, June 2009.

[21] Serafim Batzoglou, David B. Jaffe, Ken Stanley, Jonathan Butler, Sante Gnerre, Evan Mauceli, Bonnie Berger, Jill P. Mesirov, and Eric S. Lander. ARACHNE : a whole-genome shotgun assembler. *Genome Res*, 12(1) :177–189, January 2002.

[22] W. Ben-Ameur. Between fully dynamic routing and robust stable routing. In *Design and Reliable Communication Networks, 2007. DRCN 2007. 6th International Workshop on*, pages 1–6. IEEE, October 2007.

[23] Walid Ben-Ameur and Hervé Kerivin. Routing of Uncertain Traffic Demands. *Optimization and Engineering*, 6(3) :283–313, September 2005.

[24] Walid Ben-Ameur and Mateusz Żotkiewicz. Robust routing and optimal partitioning of a traffic demand polytope. *International Transactions in Operational Research*, 18(3) :307–333, May 2011.

[25] Dennis A. Benson, Mark S. Boguski, David J. Lipman, and James Ostell. GenBank. *Nucleic Acids Research*, 25(1) :1–6, January 1997.

[26] David R. Bentley, Shankar Balasubramanian, Harold P. Swerdlow, Geoffrey P. Smith, John Milton, Clive G. Brown, Kevin P. Hall, Dirk J. Evers, Colin L. Barnes, Helen R. Bignell, Jonathan M. Boutell, Jason Bryant, Richard J. Carter, R. Keira Cheetham, Anthony J. Cox, Darren J. Ellis, Michael R. Flatbush, Niall A. Gormley, Sean J. Humphray, Leslie J. Irving, Mirian S. Karbelashvili, Scott M. Kirk, Heng Li, Xiaohai Liu, Klaus S. Maisinger, Lisa J. Murray, Bojan Obradovic, Tobias Ost, Michael L. Parkinson, Mark R. Pratt, Isabelle M. J. Rasolonjatovo, Mark T. Reed, Roberto Rigatti, Chiara Rodighiero,

Mark T. Ross, Andrea Sabot, Subramanian V. Sankar, Aylwyn Scally, Gary P. Schroth, Mark E. Smith, Vincent P. Smith, Anastassia Spiridou, Peta E. Torrance, Svilen S. Tzonev, Eric H. Vermaas, Klaudia Walter, Xiaolin Wu, Lu Zhang, Mohammed D. Alam, Carole Anastasi, Ify C. Aniebo, David M. D. Bailey, Iain R. Bancarz, Saibal Banerjee, Selena G. Barbour, Primo A. Baybayan, Vincent A. Benoit, Kevin F. Benson, Claire Bevis, Phillip J. Black, Asha Boodhun, Joe S. Brennan, John A. Bridgham, Rob C. Brown, Andrew A. Brown, Dale H. Buermann, Abass A. Bundu, James C. Burrows, Nigel P. Carter, Nestor Castillo, Maria Chiara, Simon Chang, R. Neil Cooley, Natasha R. Crake, Olubunmi O. Dada, Konstantinos D. Diakoumakos, Belen D. Fernandez, David J. Earnshaw, Ugonna C. Egbujor, David W. Elmore, Sergey S. Etchin, Mark R. Ewan, Milan Fedurco, Louise J. Fraser, Karin V. Fuentes Fajardo, W. Scott Furey, David George, Kimberley J. Gietzen, Colin P. Goddard, George S. Golda, Philip A. Granieri, David E. Green, David L. Gustafson, Nancy F. Hansen, Kevin Harnish, Christian D. Haudenschild, Narinder I. Heyer, Matthew M. Hims, Johnny T. Ho, Adrian M. Horgan, Katya Hoschler, Steve Hurwitz, Denis V. Ivanov, Maria Q. Johnson, Terena James, T. A. Huw Jones, Gyoung D. Kang, Tzvetana H. Kerelska, Alan D. Kersey, Irina Khrebtukova, Alex P. Kindwall, Zoya Kingsbury, Paula I. Kokko Gonzales, Anil Kumar, Marc A. Laurent, Cynthia T. Lawley, Sarah E. Lee, Xavier Lee, Arnold K. Liao, Jennifer A. Loch, Mitch Lok, Shujun Luo, Radhika M. Mammen, John W. Martin, Patrick G. McCauley, Paul McNitt, Parul Mehta, Keith W. Moon, Joe W. Mullens, Taksina Newington, Zemin Ning, Bee L. Ng, Sonia M. Novo, Michael J. O'Neill, Mark A. Osborne, Andrew Osnowski, Omead Ostadan, Lambros L. Paraschos, Lea Pickering, Andrew C. Pike, Alger C. Pike, D. Chris Pinkard, Daniel P. Pliskin, Joe Podhasky, Victor J. Quijano, Come Raczy, Vicki H. Rae, Stephen R. Rawlings, Ana C. Rodriguez, Phyllida M. Roe, John Rogers, Maria C. Rogert Bacigalupo, Nikolai Romanov, Anthony Romieu, Rithy K. Roth, Natalie J. Rourke, Silke T. Ruediger, Eli Rusman, Raquel M. Sanches Kuiper, Martin R. Schenker, Josefina M. Seoane, Richard J. Shaw, Mitch K. Shiver, Steven W. Short, Ning L. Sizto, Johannes P. Sluis, Melanie A. Smith, Jean Ernest Sohna Sohna, Eric J. Spence, Kim Stevens, Neil Sutton, Lukasz Szajkowski, Carolyn L. Tregidgo, Gerardo Turcatti, Stephanie Vandevondele, Yuli Verhovsky, Selene M. Virk, Suzanne Wakelin, Gregory C. Walcott, Jingwen Wang, Graham J. Worsley, Juying Yan, Ling Yau, Mike Zuerlein, Jane Rogers, James C. Mullikin, Matthew E. Hurles, Nick J. McCooke, John S. West, Frank L. Oaks, Peter L. Lundberg, David Klenerman, Richard Durbin, and Anthony J. Smith. Accurate whole human genome sequencing using reversible terminator chemistry. *Nature*, 456(7218) :53–59, November 2008.

[27] Inanc Birol, Anthony Raymond, Shaun D. Jackman, Stephen Pleasance, Robin Coope, Greg A. Taylor, Macaire M. Yuen, Christopher I. Keeling, Dana Brand, Benjamin P. Vandervalk, Heather Kirk, Pawan Pandoh, Richard A. Moore, Yongjun Zhao, Andrew J.

Mungall, Barry Jaquish, Alvin Yanchuk, Carol Ritland, Brian Boyle, Jean Bousquet, Kermit Ritland, John MacKay, Jörg Bohlmann, and Steven J. M. Jones. Assembling the 20 Gb white spruce (Picea glauca) genome from whole-genome shotgun sequencing data. *Bioinformatics*, 29(12) :1492–1497, June 2013.

[28] Daniel Blankenberg, Gregory V. Kuster, Nathaniel Coraor, Guruprasad Ananda, Ross Lazarus, Mary Mangan, Anton Nekrutenko, and James Taylor. Galaxy : A Web-Based Genome Analysis Tool for Experimentalists. *Current protocols in molecular biology / edited by Frederick M. Ausubel ... [et al.]*, Chapter 19, January 2001.

[29] Marten Boetzer, Christiaan V. Henkel, Hans J. Jansen, Derek Butler, and Walter Pirovano. Scaffolding pre-assembled contigs using SSPACE. *Bioinformatics*, 27(4) :578–579, February 2011.

[30] Sébastien Boisvert, François Laviolette, and Jacques Corbeil. Ray : simultaneous assembly of reads from a mix of high-throughput sequencing technologies. *Journal of Computational Biology*, 17(11) :1519–1533, November 2010.

[31] Sebastien Boisvert, Mario Marchand, Francois Laviolette, and Jacques Corbeil. HIV-1 coreceptor usage prediction without multiple alignments : an application of string kernels. *Retrovirology*, 5(1) :110, December 2008.

[32] Sebastien Boisvert, Frederic Raymond, Elenie Godzaridis, Francois Laviolette, and Jacques Corbeil. Ray Meta : scalable de novo metagenome assembly and profiling. *Genome Biology*, 13(12) :R122+, December 2012.

[33] Keith R. Bradnam, Joseph N. Fass, Anton Alexandrov, Paul Baranay, Michael Bechner, Inanç Birol, Sébastien Boisvert, Jarrod A. Chapman, Guillaume Chapuis, Rayan Chikhi, Hamidreza Chitsaz, Wen-Chi C. Chou, Jacques Corbeil, Cristian Del Fabbro, Roderick R. Docking, Richard Durbin, Dent Earl, Scott Emrich, Pavel Fedotov, Nuno A. Fonseca, Ganeshkumar Ganapathy, Richard A. Gibbs, Sante Gnerre, Elénie Godzaridis, Steve Goldstein, Matthias Haimel, Giles Hall, David Haussler, Joseph B. Hiatt, Isaac Y. Ho, Jason Howard, Martin Hunt, Shaun D. Jackman, David B. Jaffe, Erich Jarvis, Huaiyang Jiang, Sergey Kazakov, Paul J. Kersey, Jacob O. Kitzman, James R. Knight, Sergey Koren, Tak-Wah W. Lam, Dominique Lavenier, François Laviolette, Yingrui Li, Zhenyu Li, Binghang Liu, Yue Liu, Ruibang Luo, Iain Maccallum, Matthew D. Macmanes, Nicolas Maillet, Sergey Melnikov, Delphine Naquin, Zemin Ning, Thomas D. Otto, Benedict Paten, Octávio S. Paulo, Adam M. Phillippy, Francisco Pina-Martins, Michael Place, Dariusz Przybylski, Xiang Qin, Carson Qu, Filipe J. Ribeiro, Stephen Richards, Daniel S. Rokhsar, Graham G. Ruby, Simone Scalabrin, Michael C. Schatz, David C. Schwartz, Alexey Sergushichev, Ted Sharpe, Timothy I. Shaw, Jay Shendure,

Yujian Shi, Jared T. Simpson, Henry Song, Fedor Tsarev, Francesco Vezzi, Riccardo Vicedomini, Bruno M. Vieira, Jun Wang, Kim C. Worley, Shuangye Yin, Siu-Ming M. Yiu, Jianying Yuan, Guojie Zhang, Hao Zhang, Shiguo Zhou, and Ian F. Korf. Assemblathon 2 : evaluating de novo methods of genome assembly in three vertebrate species. *GigaScience*, 2(1) :10+, July 2013.

[34] Keith R. Bradnam, Joseph N. Fass, Anton Alexandrov, Paul Baranay, Michael Bechner, İnanç Birol, Sébastien Boisvert10, Jarrod A. Chapman, Guillaume Chapuis, Rayan Chikhi, Hamidreza Chitsaz, Wen-Chi Chou, Jacques Corbeil, Cristian Del Fabbro, T. Roderick Docking, Richard Durbin, Dent Earl, Scott Emrich, Pavel Fedotov, Nuno A. Fonseca, Ganeshkumar Ganapathy, Richard A. Gibbs, Sante Gnerre, Élénie Godzaridis, Steve Goldstein, Matthias Haimel, Giles Hall, David Haussler, Joseph B. Hiatt, Isaac Y. Ho, Jason Howard, Martin Hunt, Shaun D. Jackman, David B. Jaffe, Erich Jarvis, Huaiyang Jiang, Sergey Kazakov, Paul J. Kersey, Jacob O. Kitzman, James R. Knight, Sergey Koren, Tak-Wah Lam, Dominique Lavenier, François Laviolette, Yingrui Li, Zhenyu Li, Binghang Liu, Yue Liu, Ruibang Luo, Iain MacCallum, Matthew D. MacManes, Nicolas Maillet, Sergey Melnikov, Bruno M. Vieira, Delphine Naquin, Zemin Ning, Thomas D. Otto, Benedict Paten, Octávio S. Paulo, Adam M. Phillippy, Francisco Pina-Martins, Michael Place, Dariusz Przybylski, Xiang Qin, Carson Qu, Filipe J. Ribeiro, Stephen Richards, Daniel S. Rokhsar, J. Graham Ruby, Simone Scalabrin, Michael C. Schatz, David C. Schwartz, Alexey Sergushichev, Ted Sharpe, Timothy I. Shaw, Jay Shendure, Yujian Shi, Jared T. Simpson, Henry Song, Fedor Tsarev, Francesco Vezzi, Riccardo Vicedomini, Jun Wang, Kim C. Worley, Shuangye Yin, Siu-Ming Yiu, Jianying Yuan, Guojie Zhang, Hao Zhang, Shiguo Zhou, and Ian F. Korf1. Assemblathon 2 : evaluating de novo methods of genome assembly in three vertebrate species, January 2013.

[35] Arthur Brady and Steven L. Salzberg. Phymm and PhymmBL : metagenomic phylogenetic classification with interpolated Markov models. *Nature Methods*, 6(9) :673–676, September 2009.

[36] Daniel Branton, David W. Deamer, Andre Marziali, Hagan Bayley, Steven A. Benner, Thomas Butler, Massimiliano Di Ventra, Slaven Garaj, Andrew Hibbs, Xiaohua Huang, Stevan B. Jovanovich, Predrag S. Krstic, Stuart Lindsay, Xinsheng S. Ling, Carlos H. Mastrangelo, Amit Meller, John S. Oliver, Yuriy V. Pershin, J. Michael Ramsey, Robert Riehn, Gautam V. Soni, Vincent Tabard-Cossa, Meni Wanunu, Matthew Wiggin, and Jeffery A. Schloss. The potential and challenges of nanopore sequencing. *Nature Biotechnology*, 26(10) :1146–1153, October 2008.

[37] Sydney Brenner. Sequences and consequences. *Philosophical Transactions of the Royal Society B : Biological Sciences*, 365(1537) :207–212, January 2010.

[38] Douglas W. Bryant, Weng K. Wong, and Todd C. Mockler. QSRA : a quality-value guided de novo short read assembler. *BMC Bioinformatics*, 10 :69, 2009.

[39] Jonathan Butler, Iain MacCallum, Michael Kleber, Ilya A. Shlyakhter, Matthew K. Belmonte, Eric S. Lander, Chad Nusbaum, and David B. Jaffe. ALLPATHS : de novo assembly of whole-genome shotgun microreads. *Genome Res*, 18(5) :810–820, May 2008.

[40] Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nilakantan, Arild Skjolsvold, Sam McKelvie, Yikang Xu, Shashwat Srivastav, Jiesheng Wu, Huseyin Simitci, Jaidev Haridas, Chakravarthy Uddaraju, Hemal Khatri, Andrew Edwards, Vaman Bedekar, Shane Mainali, Rafay Abbasi, Arpit Agarwal, Mian Fahim ul Haq, Muhammad Ikram ul Haq, Deepali Bhardwaj, Sowmya Dayanand, Anitha Adusumilli, Marvin McNett, Sriram Sankaran, Kavitha Manivannan, and Leonidas Rigas. Windows Azure Storage : a highly available cloud storage service with strong consistency. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 143–157, New York, NY, USA, 2011. ACM.

[41] Evelyn Camon, Michele Magrane, Daniel Barrell, Vivian Lee, Emily Dimmer, John Maslen, David Binns, Nicola Harte, Rodrigo Lopez, and Rolf Apweiler. The Gene Ontology Annotation (GOA) Database : sharing knowledge in Uniprot with Gene Ontology. *Nucleic Acids Research*, 32(suppl 1) :D262–D266, January 2004.

[42] J. Gregory Caporaso, Justin Kuczynski, Jesse Stombaugh, Kyle Bittinger, Frederic D. Bushman, Elizabeth K. Costello, Noah Fierer, Antonio G. Pena, Julia K. Goodrich, Jeffrey I. Gordon, Gavin A. Huttley, Scott T. Kelley, Dan Knights, Jeremy E. Koenig, Ruth E. Ley, Catherine A. Lozupone, Daniel McDonald, Brian D. Muegge, Meg Pirrung, Jens Reeder, Joel R. Sevinsky, Peter J. Turnbaugh, William A. Walters, Jeremy Widmann, Tanya Yatsunenko, Jesse Zaneveld, and Rob Knight. QIIME allows analysis of high-throughput community sequencing data. *Nature Methods*, 7(5) :335–336, May 2010.

[43] Rick Cattell. Scalable SQL and NoSQL Data Stores. *SIGMOD Rec.*, 39(4) :12–27, May 2011.

[44] P. S. G. Chain, D. V. Grafham, R. S. Fulton, M. G. FitzGerald, J. Hostetler, D. Muzny, J. Ali, B. Birren, D. C. Bruce, C. Buhay, J. R. Cole, Y. Ding, S. Dugan, D. Field, G. M. Garrity, R. Gibbs, T. Graves, C. S. Han, S. H. Harrison, S. Highlander, P. Hugenholtz, H. M. Khouri, C. D. Kodira, E. Kolker, N. C. Kyrpides, D. Lang, A. Lapidus, S. A. Malfatti, V. Markowitz, T. Metha, K. E. Nelson, J. Parkhill, S. Pitluck, X. Qin, T. D. Read, J. Schmutz, S. Sozhamannan, P. Sterk, R. L. Strausberg, G. Sutton, N. R. Thomson,

J. M. Tiedje, G. Weinstock, A. Wollam, Genomic Standards Consortium Human Microbiome Project Jumpstart Consortium, and J. C. Detter. Genome Project Standards in a New Era of Sequencing. *Science*, 326(5950) :236–237, October 2009.

[45] Mark Chaisson. *Combinatorial methods in computational genomics : mammalian phylogenetics using microinversions and fragment assembly with short reads*. PhD thesis, University of California, San Diego, January 2008.

[46] Mark Chaisson, Pavel Pevzner, and Haixu Tang. Fragment assembly with short reads. *Bioinformatics*, 20(13) :2067–2074, September 2004.

[47] Mark Chaisson and Glenn Tesler. Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR) : application and theory. *BMC Bioinformatics*, 13(1) :238+, 2012.

[48] Mark J. Chaisson, Dumitru Brinza, and Pavel A. Pevzner. De novo fragment assembly with short mate-paired reads : Does the read length matter ? *Genome Research*, 19(2) :336–346, 2009.

[49] Mark J. Chaisson and Pavel A. Pevzner. Short read fragment assembly of bacterial genomes. *Genome Research*, 18(2) :324–330, 2008.

[50] P. Chapdelaine, C. Pichavant, J. Rousseau, F. Paques, and J. P. Tremblay. Meganucleases can restore the reading frame of a mutated dystrophin. *Gene Therapy*, aop(current), April 2010.

[51] Dong Chen, Noel A. Eisley, Philip Heidelberger, Robert M. Senger, Yutaka Sugawara, Sameer Kumar, Valentina Salapura, David L. Satterfield, Burkhard S. Burow, and Jeffrey J. Parker. The IBM Blue Gene/Q interconnection network and message unit. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, New York, NY, USA, 2011. ACM.

[52] Bastien Chevreux, Thomas Pfisterer, Bernd Drescher, Albert J. Driesel, Werner E. G. Müller, Thomas Wetter, and Sándor Suhai. Using the miraEST assembler for reliable and automated mRNA transcript assembly and SNP detection in sequenced ESTs. *Genome Res*, 14(6) :1147–1159, June 2004.

[53] Rayan Chikhi. *Computational Methods for de novo Assembly of Next-Generation Genome Sequencing Data*. PhD thesis, École normale supérieure de Cachan - Antenne de Bretagne, July 2012.

[54] Ilseung Cho and Martin J. Blaser. The human microbiome : at the interface of health and disease. *Nature Reviews Genetics*, 13(4) :260–270, March 2012.

[55] George M. Church and Stephen Kieffer-Higgins. Multiplex DNA sequencing. *Science*, 240(4849) :185–188, April 1988.

[56] Peter J. A. Cock, Christopher J. Fields, Naohisa Goto, Michael L. Heuer, and Peter M. Rice. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, 38(6) :1767–1771, April 2010.

[57] Adriano C. Coelho, Sébastien Boisvert, Angana Mukherjee, Philippe Leprohon, Jacques Corbeil, and Marc Ouellette. Multiple Mutations in Heterogeneous Miltefosine-Resistant Leishmania major Population as Determined by Whole Genome Sequencing. *PLoS Negl Trop Dis*, 6(2) :e1512+, February 2012.

[58] Phillip E. C. Compeau, Pavel A. Pevzner, and Glenn Tesler. How to apply de Bruijn graphs to genome assembly. *Nature Biotechnology*, 29(11) :987–991, November 2011.

[59] The Human Microbiome Project Consortium. Structure, function and diversity of the healthy human microbiome. *Nature*, 486(7402) :207–214, June 2012.

[60] Thomas C. Conway and Andrew J. Bromage. Succinct data structures for assembling large genomes. *Bioinformatics (Oxford, England)*, 27(4) :479–486, February 2011.

[61] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20 :273–297, 1995.

[62] Elizabeth K. Costello, Christian L. Lauber, Micah Hamady, Noah Fierer, Jeffrey I. Gordon, and Rob Knight. Bacterial Community Variation in Human Body Habitats Across Space and Time. *Science*, 326(5960) :1694–1697, December 2009.

[63] Maura Costello, Trevor J. Pugh, Timothy J. Fennell, Chip Stewart, Lee Lichtenstein, James C. Meldrim, Jennifer L. Fostel, Dennis C. Friedrich, Danielle Perrin, Danielle Dionne, Sharon Kim, Stacey B. Gabriel, Eric S. Lander, Sheila Fisher, and Gad Getz. Discovery and characterization of artifactual mutations in deep coverage targeted capture sequencing data due to oxidative DNA damage during sample preparation. *Nucleic acids research*, 41(6), April 2013.

[64] Matthew T. Cottrell, Liying Yu, and David L. Kirchman. Sequence and Expression Analyses of Cytophaga-Like Hydrolases in a Western Arctic Metagenomic Library and the Sargasso Sea. *Applied and Environmental Microbiology*, 71(12) :8506–8513, December 2005.

[65] Petr Danecek, Adam Auton, Goncalo Abecasis, Cornelis A. Albers, Eric Banks, Mark A. DePristo, Robert Handsaker, Gerton Lunter, Gabor Marth, Stephen T. Sherry, Gilean McVean, Richard Durbin, and 1000 Genomes Project Analysis Group. The Variant Call Format and VCFtools. *Bioinformatics*, 27(15) :btr330–2158, June 2011.

[66] N. G. de Bruijn. A Combinatorial Problem. *Koninklijke Nederlandse Akademie v. Wetenschappen*, 49 :758–764, 1946.

[67] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo : amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41(6) :205–220, October 2007.

[68] Gennady Denisov, Brian Walenz, Aaron L. Halpern, Jason Miller, Nelson Axelrod, Samuel Levy, and Granger Sutton. Consensus generation and variant detection by Celera Assembler. *Bioinformatics*, 24(8) :1035–1040, April 2008.

[69] Scott Diguistini, Nancy Liao, Darren Platt, Gordon Robertson, Michael Seidel, Simon Chan, T. Roderick Docking, Inanc Birol, Robert Holt, Martin Hirst, Elaine Mardis, Marco Marra, Richard Hamelin, Jorg Bohlmann, Colette Breuil, and Steven Jones. De novo genome sequence assembly of a filamentous fungus using Sanger, 454 and Illumina sequence data. *Genome Biol*, 10(9), September 2009.

[70] Terry Disz, Sajia Akhter, Daniel Cuevas, Robert Olson, Ross Overbeek, Veronika Vonstein, Rick Stevens, and Robert A. Edwards. Accessing the SEED genome databases via Web services API : tools for programmers. *BMC bioinformatics*, 11(1) :319+, June 2010.

[71] Philip Dixon. VEGAN, a package of R functions for community ecology. *Journal of Vegetation Science*, 14(6) :927–930, 2003.

[72] Juliane C. Dohm, Claudio Lottaz, Tatiana Borodina, and Heinz Himmelbauer. SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Res*, 17(11) :1697–1706, November 2007.

[73] Juliane C. Dohm, Claudio Lottaz, Tatiana Borodina, and Heinz Himmelbauer. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Res*, 36(16), September 2008.

[74] Kara Dolinski and David Botstein. Automating the construction of gene ontologies. *Nature Biotechnology*, 31(1) :34–35, January 2013.

[75] Andreas Doring, David Weese, Tobias Rausch, and Knut Reinert. SeqAn An efficient, generic C++ library for sequence analysis. *BMC Bioinformatics*, 9(1) :11+, 2008.

[76] Joel Dudley, Yannick Pouliot, Rong Chen, Alexander Morgan, and Atul Butte. Translational bioinformatics in the cloud : an affordable alternative. *Genome Medicine*, 2(8) :51+, 2010.

[77] Joel T. Dudley and Atul J. Butte. In silico research in the era of cloud computing. *Nature Biotechnology*, 28(11) :1181–1185, November 2010.

[78] Dent Earl, Keith Bradnam, John St John, Aaron Darling, Dawei Lin, Joseph Fass, Hung On Ken O. Yu, Vince Buffalo, Daniel R. Zerbino, Mark Diekhans, Ngan Nguyen, Pramila Nuwantha N. Ariyaratne, Wing-Kin K. Sung, Zemin Ning, Matthias Haimel, Jared T. Simpson, Nuno A. Fonseca, İnanç Birol, T. Roderick Docking, Isaac Y. Ho, Daniel S. Rokhsar, Rayan Chikhi, Dominique Lavenier, Guillaume Chapuis, Delphine Naquin, Nicolas Maillet, Michael C. Schatz, David R. Kelley, Adam M. Phillippy, Sergey Koren, Shiaw-Pyng P. Yang, Wei Wu, Wen-Chi C. Chou, Anuj Srivastava, Timothy I. Shaw, J. Graham Ruby, Peter Skewes-Cox, Miguel Betegon, Michelle T. Dimon, Victor Solovyev, Igor Seledtsov, Petr Kosarev, Denis Vorobyev, Ricardo Ramirez-Gonzalez, Richard Leggett, Dan MacLean, Fangfang Xia, Ruibang Luo, Zhenyu Li, Yinlong Xie, Binghang Liu, Sante Gnerre, Iain MacCallum, Dariusz Przybylski, Filipe J. Ribeiro, Shuangye Yin, Ted Sharpe, Giles Hall, Paul J. Kersey, Richard Durbin, Shaun D. Jackman, Jarrod A. Chapman, Xiaoqiu Huang, Joseph L. DeRisi, Mario Caccamo, Yingrui Li, David B. Jaffe, Richard E. Green, David Haussler, Ian Korf, and Benedict Paten. Assemblathon 1 : a competitive assessment of de novo short read assembly methods. *Genome research*, 21(12) :2224–2241, December 2011.

[79] Daniel R. Edelson. Smart pointers : they're smart, but they're not pointers. Technical report, Santa Cruz, CA, USA, 1992.

[80] Robert A. Edwards, Robert Olson, Terry Disz, Gordon D. Pusch, Veronika Vonstein, Rick Stevens, and Ross Overbeek. Real time metagenomics : using k-mers to annotate metagenomes. *Bioinformatics (Oxford, England)*, 28(24) :3316–3317, December 2012.

[81] John Eid, Adrian Fehr, Jeremy Gray, Khai Luong, John Lyle, Geoff Otto, Paul Peluso, David Rank, Primo Baybayan, Brad Bettman, Arkadiusz Bibillo, Keith Bjornson, Bidhan Chaudhuri, Frederick Christians, Ronald Cicero, Sonya Clark, Ravindra Dalal, Alex Dewinter, John Dixon, Mathieu Foquet, Alfred Gaertner, Paul Hardenbol, Cheryl Heiner, Kevin Hester, David Holden, Gregory Kearns, Xiangxu Kong, Ronald Kuse, Yves Lacroix, Steven Lin, Paul Lundquist, Congcong Ma, Patrick Marks, Mark Maxham, Devon Murphy, Insil Park, Thang Pham, Michael Phillips, Joy Roy, Robert Sebra, Gene Shen, Jon Sorenson, Austin Tomaney, Kevin Travers, Mark Trulson, John Vieceli, Jeffrey Wegener, Dawn Wu, Alicia Yang, Denis Zaccarin, Peter Zhao, Frank Zhong, Jonas Korlach, and Stephen Turner. Real-time DNA sequencing from single polymerase molecules. *Science*, 323(5910) :133–138, January 2009.

[82] Greg Faanes, Abdulla Bataineh, Duncan Roweth, Tom Court, Edwin Froese, Bob Alverson, Tim Johnson, Joe Kopnick, Mike Higgins, and James Reinhard. Cray cascade : a

scalable HPC system based on a Dragonfly network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[83] Jian-Bing Fan, Mark S. Chee, and Kevin L. Gunderson. Highly parallel genomic assays. *Nature Reviews Genetics*, 7(8) :632–644, August 2006.

[84] R. D. Fleischmann, M. D. Adams, O. White, R. A. Clayton, E. F. Kirkness, A. R. Kerlavage, C. J. Bult, J. F. Tomb, B. A. Dougherty, and J. M. Merrick. Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. *Science*, 269(5223) :496–512, July 1995.

[85] Paul Flicek and Ewan Birney. Sense from sequence reads : methods for alignment and assembly. *Nature Methods*, 6(11 Suppl) :S6–S12, November 2009.

[86] Message P. Forum. MPI : A Message-Passing Interface Standard Version 2.2 , September 2009. Chapter author for Collective Communication and Process Topologies.

[87] Message Passing Interface Forum. MPI : A Message-Passing Interface Standard Version 3.0, 2012.

[88] Ian Foster and Carl Kesselman, editors. *The grid : blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[89] Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid : Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3) :200–222, August 2001.

[90] Carl W. Fuller, Lyle R. Middendorf, Steven A. Benner, George M. Church, Timothy Harris, Xiaohua Huang, Stevan B. Jovanovich, John R. Nelson, Jeffery A. Schloss, David C. Schwartz, and Dmitri V. Vezenov. The challenges of sequencing by synthesis. *Nature Biotechnology*, 27(11) :1013–1023, November 2009.

[91] Edgar Gabriel, Graham Fagg, George Bosilca, Thara Angskun, Jack Dongarra, Jeffrey Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph Castain, David Daniel, Richard Graham, Timothy Woodall, Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall. Open MPI : Goals, Concept, and Design of a Next Generation MPI Implementation. In Dieter Kranzlmüller, Péter Kacsuk, and Jack Dongarra, editors, *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, volume 3241 of *Lecture Notes in Computer Science*, chapter 19, pages 97–104. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[92] J. K. Gallant. The complexity of the overlap method for sequencing biopolymers. *J Theor Biol*, 101(1) :1–17, March 1983.

[93] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns : elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[94] Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Kiem-phong Vo. A Technique for Drawing Directed Graphs. In *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, pages 214–230, 1993.

[95] Steven R. Gill, Mihai Pop, Robert T. Deboy, Paul B. Eckburg, Peter J. Turnbaugh, Buck S. Samuel, Jeffrey I. Gordon, David A. Relman, Claire M. Fraser-Liggett, and Karen E. Nelson. Metagenomic analysis of the human distal gut microbiome. *Science*, 312(5778) :1355–1359, June 2006.

[96] Sante Gnerre, Iain MacCallum, Dariusz Przybylski, Filipe J. Ribeiro, Joshua N. Burton, Bruce J. Walker, Ted Sharpe, Giles Hall, Terrance P. Shea, Sean Sykes, Aaron M. Berlin, Daniel Aird, Maura Costello, Riza Daza, Louise Williams, Robert Nicol, Andreas Gnirke, Chad Nusbaum, Eric S. Lander, and David B. Jaffe. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proceedings of the National Academy of Sciences*, 108(4) :1513–1518, January 2011.

[97] Jeremy Goecks, Anton Nekrutenko, James Taylor, and The Galaxy Team. Galaxy : a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8) :R86+, August 2010.

[98] Susanne M. D. Goldberg, Justin Johnson, Dana Busam, Tamara Feldblyum, Steve Ferriera, Robert Friedman, Aaron Halpern, Hoda Khouri, Saul A. Kravitz, Federico M. Lauro, Kelvin Li, Yu H. Rogers, Robert Strausberg, Granger Sutton, Luke Tallon, Torsten Thomas, Eli Venter, Marvin Frazier, and J. Craig Venter. A Sanger/pyrosequencing hybrid approach for the generation of high-quality draft assemblies of marine microbial genomes. *Proc Natl Acad Sci U S A*, 103(30) :11240–11245, July 2006.

[99] Benjamin Good and Andrew Su. Games with a scientific purpose. *Genome Biology*, 12(12) :135+, December 2011.

[100] Phil Green. Whole-genome disassembly. *Proc Natl Acad Sci U S A*, 99(7) :4143–4144, April 2002.

[101] William Gropp. MPICH2 : A New Start for MPI Implementations. In Dieter Kranzlmüller, Jens Volkert, Peter Kacsuk, and Jack Dongarra, editors, *Recent Advances in Parallel*

*Virtual Machine and Message Passing Interface*, volume 2474 of *Lecture Notes in Computer Science*, chapter 5, pages 37–42. Springer Berlin / Heidelberg, Berlin, Heidelberg, September 2002.

[102] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI : Portable Parallel Programming with the Message Passing Interface.* MIT Press, 1994.

[103] Yan Guo, Jiang Li, Chung I. Li, Jirong Long, David Samuels, and Yu Shyr. The effect of strand bias in Illumina short-read sequencing data. *BMC Genomics*, 13(1) :666+, 2012.

[104] Alexey Gurevich, Vladislav Saveliev, Nikolay Vyahhi, and Glenn Tesler. QUAST : quality assessment tool for genome assemblies. *Bioinformatics*, 29(8) :1072–1075, April 2013.

[105] J. Handelsman, M. R. Rondon, S. F. Brady, J. Clardy, and R. M. Goodman. Molecular biological access to the chemistry of unknown soil microbes : a new frontier for natural products. *Chemistry & biology*, 5(10), October 1998.

[106] Jo Handelsman. Metagenomics : Application of Genomics to Uncultured Microorganisms. *Microbiology and Molecular Biology Reviews*, 68(4) :669–685, December 2004.

[107] Timothy D. Harris, Phillip R. Buzby, Hazen Babcock, Eric Beer, Jayson Bowers, Ido Braslavsky, Marie Causey, Jennifer Colonell, James Dimeo, J. William Efcavitch, Eldar Giladi, Jaime Gill, John Healy, Mirna Jarosz, Dan Lapen, Keith Moulton, Stephen R. Quake, Kathleen Steinmann, Edward Thayer, Anastasia Tyurina, Rebecca Ward, Howard Weiss, and Zheng Xie. Single-molecule DNA sequencing of a viral genome. *Science*, 320(5872) :106–109, April 2008.

[108] Christopher S. Henry, Matthew DeJongh, Aaron A. Best, Paul M. Frybarger, Ben Linsay, and Rick L. Stevens. High-throughput generation, optimization and analysis of genome-scale metabolic models. *Nat Biotech*, 28(9) :977–982, September 2010.

[109] Jorrit N. Herder, Herbert Bos, Ben Gras, Philip Homburg, and Andrew S. Tanenbaum. MINIX 3 : a highly reliable, self-repairing operating system. *SIGOPS Oper. Syst. Rev.*, 40(3) :80–89, July 2006.

[110] David Hernandez, Patrice François, Laurent Farinelli, Magne Osterås, and Jacques Schrenzel. De novo bacterial genome sequencing : millions of very short reads assembled on a desktop computer. *Genome Res*, 18(5) :802–809, May 2008.

[111] Michael A. Heroux, Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, Richard B. Lehoucq, Kevin R. Long, Roger P. Pawlowski, Eric T.

Phipps, Andrew G. Salinger, Heidi K. Thornquist, Ray S. Tuminaro, James M. Willenbring, Alan Williams, and Kendall S. Stanley. An overview of the Trilinos project. *ACM Trans. Math. Softw.*, 31(3) :397–423, 2005.

[112] Carl Hewitt, Peter Bishop, and Richard Steiger. A universal modular ACTOR formalism for artificial intelligence. In *Proceedings of the 3rd international joint conference on Artificial intelligence*, IJCAI'73, pages 235–245, San Francisco, CA, USA, 1973. Morgan Kaufmann Publishers Inc.

[113] Joseph B. Hiatt, Rupali P. Patwardhan, Emily H. Turner, Choli Lee, and Jay Shendure. Parallel, tag-directed assembly of locally derived short sequence reads. *Nat Meth*, 7(2) :119–122, February 2010.

[114] Saigo Hiroto, Jean P. Vert, Ueda Nobuhisa, and Akutsu Tatsuya. Local alignment kernels for biological sequences. In Tsuda and J. P. Vert, editors, *Kernel Methods in Computational Biology*, pages 131–154. MIT Press, 2004.

[115] Torsten Hoefler, James Dinan, Darius Buntinas, Pavan Balaji, Brian Barrett, Ron Brightwell, William Gropp, Vivek Kale, and Rajeev Thakur. MPI + MPI : a new hybrid approach to parallel programming with MPI plus shared memory. pages 1–16, 2013.

[116] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns : Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[117] Robert A. Holt and Steven J. M. Jones. The new paradigm of flow cell sequencing. *Genome Research*, 18(6) :839–846, 2008.

[118] Mohammad S. Hossain, Navid Azimi, and Steven Skiena. Crystallizing short-read assemblies around seeds. *BMC Bioinformatics*, 10 Suppl 1, 2009.

[119] Adina C. Howe, Jason Pell, Rosangela Canino-Koning, Rachel Mackelprang, Susannah Tringe, Janet Jansson, James M. Tiedje, and C. Titus Brown. Illumina Sequencing Artifacts Revealed by Connectivity Analysis of Metagenomic Datasets. *arXiv*, December 2012.

[120] Martin Hunt, Taisei Kikuchi, Mandy Sanders, Chris Newbold, Matthew Berriman, and Thomas Otto. REAPR : a universal tool for genome assembly evaluation. *Genome Biology*, 14(5) :R47+, May 2013.

[121] D. H. Huson, K. Reinert, S. A. Kravitz, K. A. Remington, A. L. Delcher, I. M. Dew, M. Flanigan, A. L. Halpern, Z. Lai, C. M. Mobarry, G. G. Sutton, and E. W. Myers. Design of a compartmentalized shotgun assembler for the human genome. *Bioinformatics*, 17 Suppl 1 :S132–S139, 2001.

[122] Daniel H. Huson, Suparna Mitra, Hans-Joachim Ruscheweyh, Nico Weber, and Stephan C. Schuster. Integrative analysis of environmental sequences using MEGAN4. *Genome Research*, 21(9) :1552–1560, September 2011.

[123] Daniel H. Huson, Knut Reinert, and Eugene W. Myers. The greedy path-merging algorithm for contig scaffolding. *J. ACM*, 49(5) :603–615, 2002.

[124] G. Hutchinson. Evaluation of polymer sequence fragment data using graph theory. *Bulletin of Mathematical Biophysics*, 31(3) :541–562, 1969.

[125] R. M. Idury and M. S. Waterman. A new algorithm for DNA sequence assembly. *Journal of Computational Biology*, 2(2) :291–306, 1995.

[126] Zamin Iqbal, Mario Caccamo, Isaac Turner, Paul Flicek, and Gil McVean. De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nature Genetics*, 44(2) :226–232, January 2012.

[127] Shaun Jackman and Inanc Birol. Assembling genomes using short-read sequencing technology. *Genome Biology*, 11(1) :202+, 2010.

[128] Benjamin G. Jackson, Patrick S. Schnable, and Srinivas Aluru. Parallel short sequence assembly of transcriptomes. *BMC bioinformatics*, 10 Suppl 1(Suppl 1) :S14+, 2009.

[129] BenjaminG Jackson, PatrickS Schnable, and Srinivas Aluru. Assembly of Large Genomes from Paired Short Reads. In Sanguthevar Rajasekaran, editor, *Bioinformatics and Computational Biology*, volume 5462 of *Lecture Notes in Computer Science*, pages 30–43. Springer Berlin Heidelberg, 2009.

[130] William R. Jeck, Josephine A. Reinhardt, David A. Baltrus, Matthew T. Hickenbotham, Vincent Magrini, Elaine R. Mardis, Jeffery L. Dangl, and Corbin D. Jones. Extending assembly of short DNA sequences to handle error. *Bioinformatics*, 23(21) :2942–2944, November 2007.

[131] Mark A. Jensen, Mia Coetzer, Angélique B. van 't Wout, Lynn Morris, and James I. Mullins. A reliable phenotype predictor for human immunodeficiency virus type 1 subtype C based on envelope V3 sequences. *J Virol*, 80(10) :4698–4704, May 2006.

[132] T. Joachims. Making large-Scale SVM Learning Practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[133] Sebastian Junemann, Fritz J. Sedlazeck, Karola Prior, Andreas Albersmeier, Uwe John, Jorn Kalinowski, Alexander Mellmann, Alexander Goesmann, Arndt von Haeseler, Jens Stoye, and Dag Harmsen. Updating benchtop sequencing performance comparison. *Nat Biotech*, 31(4) :294–296, April 2013.

[134] Laxmikant V. Kale and Sanjeev Krishnan. CHARM++ : a portable concurrent object oriented system based on C++. In *Proceedings of the eighth annual conference on Object-oriented programming systems, languages, and applications*, OOPSLA '93, pages 91–108, New York, NY, USA, 1993. ACM.

[135] Laxmikant V. Kale and Gengbin Zheng. Charm++ and AMPI : Adaptive Runtime Strategies via Migratable Objects. pages 265–282, 2009.

[136] RichardM Karp. Reducibility Among Combinatorial Problems. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 219–241. Springer Berlin Heidelberg, 2010.

[137] Alexander Kawrykow, Gary Roumanis, Alfred Kam, Daniel Kwak, Clarence Leung, Chu Wu, Eleyine Zarour, Phylo players, Luis Sarmenta, Mathieu Blanchette, and Jérôme Waldispühl. Phylo : a citizen science approach for improving multiple sequence alignment. *PloS one*, 7(3) :e31362+, March 2012.

[138] W. James Kent. BLAT–the BLAST-like alignment tool. *Genome Res*, 12(4) :656–664, April 2002.

[139] Donald E. Knuth. *Art of Computer Programming, Volume 3 : Sorting and Searching (2nd Edition)*. Addison-Wesley Professional, 2 edition, May 1998.

[140] Konstantinos Krampis, Tim Booth, Brad Chapman, Bela Tiwari, Mesude Bicak, Dawn Field, and Karen E. Nelson. Cloud BioLinux : pre-configured and on-demand bioinformatics computing for the genomics community. *BMC bioinformatics*, 13(1) :42+, March 2012.

[141] Lutz Krause, Naryttza N. Diaz, Alexander Goesmann, Scott Kelley, Tim W. Nattkemper, Forest Rohwer, Robert A. Edwards, and Jens Stoye. Phylogenetic classification of short environmental DNA fragments. *Nucleic Acids Research*, 36(7) :2230–2239, April 2008.

[142] Tamara Kulikova, Philippe Aldebert, Nicola Althorpe, Wendy Baker, Kirsty Bates, Paul Browne, Alexandra van den Broek, Guy Cochrane, Karyn Duggan, Ruth Eberhardt, Nadeem Faruque, Maria Garcia-Pastor, Nicola Harte, Carola Kanz, Rasko Leinonen, Quan Lin, Vincent Lombard, Rodrigo Lopez, Renato Mancuso, Michelle McHale, Francesco Nardone, Ville Silventoinen, Peter Stoehr, Guenter Stoesser, Mary Ann, Katerina Tzouvara, Robert Vaughan, Dan Wu, Weimin Zhu, and Rolf Apweiler. The EMBL Nucleotide Sequence Database. *Nucleic Acids Research*, 32(suppl 1) :D27–D30, January 2004.

[143] Stefan Kurtz, Adam Phillippy, Arthur L. Delcher, Michael Smoot, Martin Shumway, Corina Antonescu, and Steven L. Salzberg. Versatile and open software for comparing large genomes. *Genome Biol*, 5(2), 2004.

[144] E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, K. Dewar, M. Doyle, W. FitzHugh, R. Funke, D. Gage, K. Harris, A. Heaford, J. Howland, L. Kann, J. Lehoczky, R. LeVine, P. McEwan, K. McKernan, J. Meldrim, J. P. Mesirov, C. Miranda, W. Morris, J. Naylor, C. Raymond, M. Rosetti, R. Santos, A. Sheridan, C. Sougnez, N. Stange Thomann, N. Stojanovic, A. Subramanian, D. Wyman, J. Rogers, J. Sulston, R. Ainscough, S. Beck, D. Bentley, J. Burton, C. Clee, N. Carter, A. Coulson, R. Deadman, P. Deloukas, A. Dunham, I. Dunham, R. Durbin, L. French, D. Grafham, S. Gregory, T. Hubbard, S. Humphray, A. Hunt, M. Jones, C. Lloyd, A. McMurray, L. Matthews, S. Mercer, S. Milne, J. C. Mullikin, A. Mungall, R. Plumb, M. Ross, R. Shownkeen, S. Sims, R. H. Waterston, R. K. Wilson, L. W. Hillier, J. D. McPherson, M. A. Marra, E. R. Mardis, L. A. Fulton, A. T. Chinwalla, K. H. Pepin, W. R. Gish, S. L. Chissoe, M. C. Wendl, K. D. Delehaunty, T. L. Miner, A. Delehaunty, J. B. Kramer, L. L. Cook, R. S. Fulton, D. L. Johnson, P. J. Minx, S. W. Clifton, T. Hawkins, E. Branscomb, P. Predki, P. Richardson, S. Wenning, T. Slezak, N. Doggett, J. F. Cheng, A. Olsen, S. Lucas, C. Elkin, E. Uberbacher, M. Frazier, R. A. Gibbs, D. M. Muzny, S. E. Scherer, J. B. Bouck, E. J. Sodergren, K. C. Worley, C. M. Rives, J. H. Gorrell, M. L. Metzker, S. L. Naylor, R. S. Kucherlapati, D. L. Nelson, G. M. Weinstock, Y. Sakaki, A. Fujiyama, M. Hattori, T. Yada, A. Toyoda, T. Itoh, C. Kawagoe, H. Watanabe, Y. Totoki, T. Taylor, J. Weissenbach, R. Heilig, W. Saurin, F. Artiguenave, P. Brottier, T. Bruls, E. Pelletier, C. Robert, P. Wincker, D. R. Smith, L. Doucette Stamm, M. Rubenfield, K. Weinstock, H. M. Lee, J. Dubois, A. Rosenthal, M. Platzer, G. Nyakatura, S. Taudien, A. Rump, H. Yang, J. Yu, J. Wang, G. Huang, J. Gu, L. Hood, L. Rowen, A. Madan, S. Qin, R. W. Davis, N. A. Federspiel, A. P. Abola, M. J. Proctor, R. M. Myers, J. Schmutz, M. Dickson, J. Grimwood, D. R. Cox, M. V. Olson, R. Kaul, C. Raymond, N. Shimizu, K. Kawasaki, S. Minoshima, G. A. Evans, M. Athanasiou, R. Schultz, B. A. Roe, F. Chen, H. Pan, J. Ramser, H. Lehrach, R. Reinhardt, W. R. McCombie, de la Bastide, N. Dedhia, H. Blöcker, K. Hornischer,

G. Nordsiek, R. Agarwala, L. Aravind, J. A. Bailey, A. Bateman, S. Batzoglou, E. Birney, P. Bork, D. G. Brown, C. B. Burge, L. Cerutti, H. C. Chen, D. Church, M. Clamp, R. R. Copley, T. Doerks, S. R. Eddy, E. E. Eichler, T. S. Furey, J. Galagan, J. G. Gilbert, C. Harmon, Y. Hayashizaki, D. Haussler, H. Hermjakob, K. Hokamp, W. Jang, L. S. Johnson, T. A. Jones, S. Kasif, A. Kaspryzk, S. Kennedy, W. J. Kent, P. Kitts, E. V. Koonin, I. Korf, D. Kulp, D. Lancet, T. M. Lowe, A. McLysaght, T. Mikkelsen, J. V. Moran, N. Mulder, V. J. Pollara, C. P. Ponting, G. Schuler, J. Schultz, G. Slater, A. F. Smit, E. Stupka, J. Szustakowski, D. Thierry Mieg, J. Thierry Mieg, L. Wagner, J. Wallis, R. Wheeler, A. Williams, Y. I. Wolf, K. H. Wolfe, S. P. Yang, R. F. Yeh, F. Collins, M. S. Guyer, J. Peterson, A. Felsenfeld, K. A. Wetterstrand, A. Patrinos, M. J. Morgan, P. de Jong, J. J. Catanese, K. Osoegawa, H. Shizuya, S. Choi, Y. J. Chen, J. Szustakowki, and International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409(6822) :860–921, February 2001.

[145] Eric S. Lander. Initial impact of the sequencing of the human genome. *Nature*, 470(7333) :187–197, February 2011.

[146] J. Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 6 :273–306, 2005.

[147] Ben Langmead and Steven L. Salzberg. Fast gapped-read alignment with Bowtie 2. *Nat Meth*, 9(4) :357–359, April 2012.

[148] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(3) :R25–10, March 2009.

[149] Jonathan Laserson, Vladimir Jojic, and Daphne Koller. Genovo : de novo assembly for metagenomes. *Journal of computational biology : a journal of computational molecular cell biology*, 18(3) :429–443, March 2011.

[150] J. K. Lawder and P. J. H. King. Querying multi-dimensional data indexed using the Hilbert space-filling curve. *SIGMOD Rec.*, 30(1) :19–24, March 2001.

[151] Ryan M. Layer, Kevin Skadron, Gabriel Robins, Ira M. Hall, and Aaron R. Quinlan. Binary Interval Search : a scalable algorithm for counting interval intersections. *Bioinformatics*, 29(1) :1–7, January 2013.

[152] Rasko Leinonen, Hideaki Sugawara, and Martin Shumway. The Sequence Read Archive. *Nucleic Acids Research*, 39(suppl 1) :D19–D21, January 2011.

[153] T. Lengauer, O. Sander, S. Sierra, A. Thielen, and R. Kaiser. Bioinformatics prediction of HIV coreceptor usage. *Nat. Biotechnol.*, 25 :1407–1410, December 2007.

[154] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel : a string kernel for SVM protein classification. *Pac Symp Biocomput*, pages 564–575, 2002.

[155] C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20 :467–476, March 2004.

[156] Ruth E. Ley, Catherine A. Lozupone, Micah Hamady, Rob Knight, and Jeffrey I. Gordon. Worlds within worlds : evolution of the vertebrate gut microbiota. *Nature Reviews Microbiology*, 6(10) :776–788, October 2008.

[157] Heng Li. Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly. *Bioinformatics*, 28(14) :1838–1844, July 2012.

[158] Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14) :1754–1760, July 2009.

[159] Heng Li and Richard Durbin. Fast and accurate long-read alignment with Burrows–Wheeler transform. *Bioinformatics*, 26(5) :589–595, March 2010.

[160] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, and 1000 Genome Project Data Processing Subgroup. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16) :2078–2079, August 2009.

[161] Heng Li and Nils Homer. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, 11(5) :473–483, September 2010.

[162] Heng Li, Jue Ruan, and Richard Durbin. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res*, 18(11) :1851–1858, November 2008.

[163] Ruiqiang Li, Chang Yu, Yingrui Li, Tak W. Lam, Siu M. Yiu, Karsten Kristiansen, and Jun Wang. SOAP2 : an improved ultrafast tool for short read alignment. *Bioinformatics*, 25(15) :1966–1967, August 2009.

[164] Doron Lipson, Tal Raz, Alix Kieu, Daniel R. Jones, Eldar Giladi, Edward Thayer, John F. Thompson, Stan Letovsky, Patrice Milos, and Marie Causey. Quantification of the yeast transcriptome by single-molecule sequencing. *Nat Biotechnol*, 27(7) :652–658, July 2009.

[165] Bo Liu, Theodore Gibbons, Mohammad Ghodsi, and Mihai Pop. MetaPhyler : Taxonomic profiling for metagenomic sequences. In *2010 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 95–100. IEEE, December 2010.

[166] Chris Loken, Daniel Gruner, Leslie Groer, Richard Peltier, Neil Bunn, Michael Craig, Teresa Henriques, Jillian Dempsey, Ching-Hsing Yu, Joseph Chen, L. Jonathan Dursi, Jason Chong, Scott Northrup, Jaime Pinto, Neil Knecht, and Ramses Van Zon. SciNet : Lessons Learned from Building a Power-efficient Top-20 System and Data Centre. *Journal of Physics : Conference Series*, 256(1) :012026+, December 2010.

[167] Nicholas J. Loman, Raju V. Misra, Timothy J. Dallman, Chrystala Constantinidou, Saheer E. Gharbia, John Wain, and Mark J. Pallen. Performance comparison of benchtop high-throughput sequencing platforms. *Nature Biotechnology*, 30(5) :434–439, April 2012.

[168] Patrick Lorenz and Jurgen Eck. Outlook : Metagenomics and industrial applications. *Nature Reviews Microbiology*, 3(6) :510–516, June 2005.

[169] Catherine Lozupone, Micah Hamady, and Rob Knight. UniFrac–an online tool for comparing microbial community diversity in a phylogenetic context. *BMC bioinformatics*, 7(1) :371+, August 2006.

[170] Catherine Lozupone and Rob Knight. UniFrac : a New Phylogenetic Method for Comparing Microbial Communities. *Applied and Environmental Microbiology*, 71(12) :8228–8235, December 2005.

[171] Catherine A. Lozupone and Rob Knight. Global patterns in bacterial diversity. *Proceedings of the National Academy of Sciences*, 104(27) :11436–11440, July 2007.

[172] Ruibang Luo, Binghang Liu, Yinlong Xie, Zhenyu Li, Weihua Huang, Jianying Yuan, Guangzhu He, Yanxiang Chen, Qi Pan, Yunjie Liu, Jingbo Tang, Gengxiong Wu, Hao Zhang, Yujian Shi, Yong Liu, Chang Yu, Bo Wang, Yao Lu, Changlei Han, David W. Cheung, Siu-Ming Yiu, Shaoliang Peng, Zhu Xiaoqian, Guangming Liu, Xiangke Liao, Yingrui Li, Huanming Yang, Jian Wang, Tak-Wah Lam, and Jun Wang. SOAPdenovo2 : an empirically improved memory-efficient short-read de novo assembler. *GigaScience*, 1(1) :18+, 2012.

[173] Tanja Magoc, Stephan Pabinger, Stefan Canzar, Xinyue Liu, Qi Su, Daniela Puiu, Luke J. Tallon, and Steven L. Salzberg. GAGE-B : an evaluation of genome assemblers for bacterial organisms. *Bioinformatics*, 29(14) :1718–1725, July 2013.

[174] Prashant Mali, Luhan Yang, Kevin M. Esvelt, John Aach, Marc Guell, James E. DiCarlo, Julie E. Norville, and George M. Church. RNA-Guided Human Genome Engineering via Cas9. *Science*, 339(6121) :823–826, February 2013.

[175] Elaine R. Mardis. New strategies and emerging technologies for massively parallel sequencing : applications in medical research. *Genome medicine*, 1(4), April 2009.

[176] Elaine R. Mardis. The $1,000 genome, the $100,000 analysis ? *Genome medicine*, 2(11) :84+, 2010.

[177] Marcel Margulies, Michael Egholm, William E. Altman, Said Attiya, Joel S. Bader, Lisa A. Bemben, Jan Berka, Michael S. Braverman, Yi-Ju Chen, Zhoutao Chen, Scott B. Dewell, Lei Du, Joseph M. Fierro, Xavier V. Gomes, Brian C. Godwin, Wen He, Scott Helgesen, Chun H. Ho, Gerard P. Irzyk, Szilveszter C. Jando, Maria L. I. Alenquer, Thomas P. Jarvie, Kshama B. Jirage, Jong-Bum Kim, James R. Knight, Janna R. Lanza, John H. Leamon, Steven M. Lefkowitz, Ming Lei, Jing Li, Kenton L. Lohman, Hong Lu, Vinod B. Makhijani, Keith E. Mcdade, Michael P. Mckenna, Eugene W. Myers, Elizabeth Nickerson, John R. Nobile, Ramona Plant, Bernard P. Puc, Michael T. Ronan, George T. Roth, Gary J. Sarkis, Jan F. Simons, John W. Simpson, Maithreyan Srinivasan, Karrie R. Tartaro, Alexander Tomasz, Kari A. Vogt, Greg A. Volkmer, Shally H. Wang, Yong Wang, Michael P. Weiner, Pengguang Yu, Richard F. Begley, and Jonathan M. Rothberg. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057) :376–380, July 2005.

[178] Vivien Marx. Biology : The big challenges of big data. *Nature*, 498(7453) :255–260, June 2013.

[179] Vivien Marx. Genomics in the clouds. *Nature Methods*, 10(10) :941–945, September 2013.

[180] Chris A. Mattmann. Computing : A vision for data science. *Nature*, 493(7433) :473–475, January 2013.

[181] A. M. Maxam and W. Gilbert. A new method for sequencing DNA. *Proc Natl Acad Sci U S A*, 74(2) :560–564, February 1977.

[182] Daniel McDonald, Morgan N. Price, Julia Goodrich, Eric P. Nawrocki, Todd Z. DeSantis, Alexander Probst, Gary L. Andersen, Rob Knight, and Philip Hugenholtz. An improved Greengenes taxonomy with explicit ranks for ecological and evolutionary analyses of bacteria and archaea. *The ISME Journal*, 6(3) :610–618, December 2011.

[183] John D. McPherson. Next-generation gap. *Nature Methods*, 6(11 Suppl) :S2–S5, November 2009.

[184] Duccio Medini, Davide Serruto, Julian Parkhill, David A. Relman, Claudio Donati, Richard Moxon, Stanley Falkow, and Rino Rappuoli. Microbiology in the post-genomic era. *Nat Rev Microbiol*, 6(6) :419–430, June 2008.

[185] Paul Medvedev and Michael Brudno. Maximum likelihood genome assembly. *J Comput Biol*, 16(8) :1101–1116, August 2009.

[186] Paul Medvedev, Konstantinos Georgiou, Gene Myers, and Michael Brudno. Computability of Models for Sequence Assembly. In *Lecture Notes in Computer Science*, pages 289–301. 2007.

[187] F. Meyer, D. Paarmann, M. D'Souza, R. Olson, E. M. Glass, M. Kubal, T. Paczian, A. Rodriguez, R. Stevens, A. Wilke, J. Wilkening, and R. A. Edwards. The metagenomics RAST server - a public resource for the automatic phylogenetic and functional analysis of metagenomes. *BMC Bioinformatics*, 9(1) :386–8, December 2008.

[188] Chase A. Miller, Jon Anthony, Michelle M. Meyer, and Gabor Marth. Scribl : an HTML5 Canvas-based graphics library for visualizing genomic data over the web. *Bioinformatics*, 29(3) :381–383, February 2013.

[189] Jason R. Miller, Arthur L. Delcher, Sergey Koren, Eli Venter, Brian P. Walenz, Anushka Brownley, Justin Johnson, Kelvin Li, Clark Mobarry, and Granger Sutton. Aggressive assembly of pyrosequencing reads with mates. *Bioinformatics*, 24(24) :2818–2824, December 2008.

[190] Jason R. Miller, Sergey Koren, and Granger Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6) :315–327, June 2010.

[191] M. J. Miller and J. I. Powell. A quantitative comparison of DNA sequence assembly programs. *Journal of computational biology : a journal of computational molecular cell biology*, 1(4) :257–269, 1994.

[192] Suparna Mitra, Paul Rupek, Daniel Richter, Tim Urich, Jack Gilbert, Folker Meyer, Andreas Wilke, and Daniel Huson. Functional analysis of metagenomes and metatranscriptomes using SEED and KEGG. *BMC Bioinformatics*, 12(Suppl 1) :S21+, 2011.

[193] Angana Mukherjee, Sébastien Boisvert, Rubens L. Monte-Neto, Adriano C. Coelho, Frederic Raymond, Rita Mukhopadhyay, Jacques Corbeil, and Marc Ouellette. Telomeric gene deletion and intrachromosomal amplification in antimony-resistant Leishmania. *Molecular Microbiology*, 88(1) :189–202, April 2013.

[194] E. W. Myers. Toward simplifying and accurately formulating fragment assembly. *Journal of computational biology : a journal of computational molecular cell biology*, 2(2) :275–290, 1995.

[195] E. W. Myers, G. G. Sutton, A. L. Delcher, I. M. Dew, D. P. Fasulo, M. J. Flanigan, S. A. Kravitz, C. M. Mobarry, K. H. Reinert, K. A. Remington, E. L. Anson, R. A. Bolanos, H. H. Chou, C. M. Jordan, A. L. Halpern, S. Lonardi, E. M. Beasley, R. C. Brandon, L. Chen, P. J. Dunn, Z. Lai, Y. Liang, D. R. Nusskern, M. Zhan, Q. Zhang, X. Zheng,

G. M. Rubin, M. D. Adams, and J. C. Venter. A whole-genome assembly of Drosophila. *Science*, 287(5461) :2196–2204, March 2000.

[196] Eugene W. Myers. The fragment assembly string graph. *Bioinformatics*, 21 Suppl 2 :ii79–ii85, September 2005.

[197] Eugene W. Myers, Granger G. Sutton, Hamilton O. Smith, Mark D. Adams, and J. Craig Venter. On the sequencing and assembly of the human genome. *Proc Natl Acad Sci U S A*, 99(7) :4145–4146, April 2002.

[198] Niranjan Nagarajan and Mihai Pop. Sequence assembly demystified. *Nat Rev Genet*, 14(3) :157–167, March 2013.

[199] Niranjan Nagarajan, Timothy D. Read, and Mihai Pop. Scaffolding and validation of bacterial genome assemblies using optical restriction maps. *Bioinformatics*, 24(10) :1229–1235, May 2008.

[200] Shota Nakamura, Norihiro Maeda, Ionut Mihai M. Miron, Myonsun Yoh, Kaori Izutsu, Chidoh Kataoka, Takeshi Honda, Teruo Yasunaga, Takaaki Nakaya, Jun Kawai, Yoshihide Hayashizaki, Toshihiro Horii, and Tetsuya Iida. Metagenomic diagnosis of bacterial infections. *Emerging infectious diseases*, 14(11) :1784–1786, November 2008.

[201] Toshiaki Namiki, Tsuyoshi Hachiya, Hideaki Tanaka, and Yasubumi Sakakibara. MetaVelvet : an extension of Velvet assembler to de novo metagenome assembly from short sequence reads. In *Proceedings of the 2nd ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, BCB '11, pages 116–124, New York, NY, USA, 2011. ACM.

[202] Toshiaki Namiki, Tsuyoshi Hachiya, Hideaki Tanaka, and Yasubumi Sakakibara. MetaVelvet : an extension of Velvet assembler to de novo metagenome assembly from short sequence reads. *Nucleic Acids Research*, 40(20) :e155, November 2012.

[203] Priya Narasingarao, Sheila Podell, Juan A. Ugalde, Celine Brochier-Armanet, Joanne B. Emerson, Jochen J. Brocks, Karla B. Heidelberg, Jillian F. Banfield, and Eric E. Allen. De novo metagenomic assembly reveals abundant novel major lineage of Archaea in hypersaline microbial communities. *The ISME Journal*, 6(1) :81–93, June 2011.

[204] R. K. Naviaux, B. Good, J. D. McPherson, D. L. Steffen, D. Markusic, B. Ransom, and J. Corbeil. Sand DNA - a genetic library of life at the water's edge. *Marine Ecology Progress Series*, 301 :9–22, 2005.

[205] Sarah B. Ng, Emily H. Turner, Peggy D. Robertson, Steven D. Flygare, Abigail W. Bigham, Choli Lee, Tristan Shaffer, Michelle Wong, Arindam Bhattacharjee, Evan E.

Eichler, Michael Bamshad, Deborah A. Nickerson, and Jay Shendure. Targeted capture and massively parallel sequencing of 12 human exomes. *Nature*, 461(7261) :272–276, September 2009.

[206] Cydney B. Nielsen, Michael Cantor, Inna Dubchak, David Gordon, and Ting Wang. Visualizing genomes : techniques and challenges. *Nature Methods*, 7(3 Suppl), March 2010.

[207] Cydney B. Nielsen, Shaun D. Jackman, Inanç Birol, and Steven J. Jones. ABySS-Explorer : visualizing genome sequence assemblies. *IEEE transactions on visualization and computer graphics*, 15(6) :881–888, October 2009.

[208] Alexei Nordell Markovits, Charles Joly Beauparlant, Dominique Toupin, Shengrui Wang, Arnaud Droit, and Nicolas Gevry. NGS++ : a library for rapid prototyping of epigenomics software tools. *Bioinformatics (Oxford, England)*, 29(15) :1893–1894, August 2013.

[209] Fatih Ozsolak and Patrice M. Milos. RNA sequencing : advances, challenges and opportunities. *Nat Rev Genet*, 12(2) :87–98, February 2011.

[210] Fatih Ozsolak, Adam R. Platt, Dan R. Jones, Jeffrey G. Reifenberger, Lauryn E. Sass, Peter McInerney, John F. Thompson, Jayson Bowers, Mirna Jarosz, and Patrice M. Milos. Direct RNA sequencing. *Nature*, 461(7265) :814–818, October 2009.

[211] Theodore R. Pak and Frederick P. Roth. ChromoZoom : a flexible, fluid, web-based genome browser. *Bioinformatics*, 29(3) :384–386, February 2013.

[212] Konrad Paszkiewicz and David J. Studholme. De novo assembly of short sequence reads. *Briefings in bioinformatics*, 11(5) :457–472, September 2010.

[213] Jason Pell, Arend Hintze, Rosangela Canino-Koning, Adina Howe, James M. Tiedje, and C. Titus Brown. Scaling metagenome sequence assembly with probabilistic de Bruijn graphs. *Proceedings of the National Academy of Sciences of the United States of America*, 109(33) :13272–13277, August 2012.

[214] Yu Peng, Henry C. M. Leung, S. M. Yiu, and Francis Y. L. Chin. Meta-IDBA : a de Novo assembler for metagenomic data. *Bioinformatics*, 27(13) :i94–i101, July 2011.

[215] Yu Peng, Henry C. M. Leung, S. M. Yiu, and Francis Y. L. Chin. IDBA-UD : a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics*, 28(11) :1420–1428, June 2012.

[216] Yu Peng, HenryC Leung, S. M. Yiu, and FrancisY Chin. IDBA – A Practical Iterative de Bruijn Graph De Novo Assembler. In Bonnie Berger, editor, *Research in Computational Molecular Biology*, volume 6044 of *Lecture Notes in Computer Science*, chapter 28, pages 426–440. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[217] P. A. Pevzner. 1-Tuple DNA sequencing : computer analysis. *J Biomol Struct Dyn*, 7(1) :63–73, August 1989.

[218] P. A. Pevzner and H. Tang. Fragment assembly with double-barreled data. *Bioinformatics*, 17 Suppl 1 :S225–S233, 2001.

[219] P. A. Pevzner, H. Tang, and M. S. Waterman. An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17) :9748–9753, 2001.

[220] Pavel Pevzner and Ron Shamir. Computing has changed biology–biology education must catch up. *Science*, 325(5940) :541–542, July 2009.

[221] Pavel A. Pevzner. Educating biologists in the 21st century : bioinformatics scientists versus bioinformatics technicians. *Bioinformatics*, 20(14) :2159–2161, September 2004.

[222] Pavel A. Pevzner, Sangtae Kim, and Julio Ng. Comment on "Protein sequences from mastodon and Tyrannosaurus rex revealed by mass spectrometry". *Science*, 321(5892), August 2008.

[223] Pavel A. Pevzner, Paul A. Pevzner, Haixu Tang, and Glenn Tesler. De novo repeat classification and fragment assembly. *Genome Res*, 14(9) :1786–1796, September 2004.

[224] C. D. Pham. Comparison of message aggregation strategies for parallel simulations on a high performance cluster. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000. Proceedings. 8th International Symposium on*, pages 358–365. IEEE, 2000.

[225] Son K. Pham, Dmitry Antipov, Alexander Sirotkin, Glenn Tesler, Pavel A. Pevzner, and Max A. Alekseyev. Pathset Graphs : A Novel Approach for Comprehensive Utilization of Paired Reads in Genome Assembly. *Journal of Computational Biology*, 20(4) :359–371, April 2013.

[226] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kalé, and Klaus Schulten. Scalable molecular dynamics with NAMD. *J. Comput. Chem.*, 26(16) :1781–1802, December 2005.

[227] Satish Pillai, Benjamin Good, Douglas Richman, and Jacques Corbeil. A new perspective on V3 phenotype prediction. *AIDS Res Hum Retroviruses*, 19(2) :145–149, February 2003.

[228] Andrew Pollack. DNA sequencing caught in deluge of data. *New York Times*, 1, 2011.

[229] Mihai Pop. Genome assembly reborn : recent computational challenges. *Brief Bioinform*, 10(4) :354–366, July 2009.

[230] Mihai Pop, Daniel S. Kosack, and Steven L. Salzberg. Hierarchical scaffolding with Bambus. *Genome Res*, 14(1) :149–159, January 2004.

[231] Dmitry Pushkarev, Norma F. Neff, and Stephen R. Quake. Single-molecule sequencing of an individual human genome. *Nat Biotech*, 27(9) :847–850, September 2009.

[232] Oliver G. Pybus and Andrew Rambaut. Evolutionary analysis of the dynamics of viral infectious disease. *Nat Rev Genet*, 10(8) :540–550, August 2009.

[233] Junjie Qin, Ruiqiang Li, Jeroen Raes, Manimozhiyan Arumugam, Kristoffer S. Burgdorf, Chaysavanh Manichanh, Trine Nielsen, Nicolas Pons, Florence Levenez, Takuji Yamada, Daniel R. Mende, Junhua Li, Junming Xu, Shaochuan Li, Dongfang Li, Jianjun Cao, Bo Wang, Huiqing Liang, Huisong Zheng, Yinlong Xie, Julien Tap, Patricia Lepage, Marcelo Bertalan, Jean-Michel Batto, Torben Hansen, Denis Le Paslier, Allan Linneberg, H. Bjorn Nielsen, Eric Pelletier, Pierre Renault, Thomas Sicheritz-Ponten, Keith Turner, Hongmei Zhu, Chang Yu, Shengting Li, Min Jian, Yan Zhou, Yingrui Li, Xiuqing Zhang, Songgang Li, Nan Qin, Huanming Yang, Jian Wang, Soren Brunak, Joel Dore, Francisco Guarner, Karsten Kristiansen, Oluf Pedersen, Julian Parkhill, Jean Weissenbach, Peer Bork, S. Dusko Ehrlich, and Jun Wang. A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, 464(7285) :59–65, March 2010.

[234] Xiaohong Qiu, Jaliya Ekanayake, Scott Beason, Thilina Gunarathne, Geoffrey Fox, Roger Barga, and Dennis Gannon. Cloud technologies for bioinformatics applications. In *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers*, MTAGS '09, New York, NY, USA, 2009. ACM.

[235] Michael Quail, Miriam Smith, Paul Coupland, Thomas Otto, Simon Harris, Thomas Connor, Anna Bertoni, Harold Swerdlow, and Yong Gu. A tale of three next generation sequencing platforms : comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. *BMC Genomics*, 13(1) :341+, July 2012.

[236] Aaron R. Quinlan and Ira M. Hall. BEDTools : a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6) :841–842, March 2010.

[237] Kim R. Rasmussen, Jens Stoye, and Eugene W. Myers. Efficient q-gram filters for finding all epsilon-matches over a given length. *Journal of computational biology : a journal of computational molecular cell biology*, 13(2) :296–308, March 2006.

[238] Frédéric Raymond, Julie Carbonneau, Nancy Boucher, Lynda Robitaille, Sébastien Boisvert, Whei-Kuo Wu, Gaston De Serres, Guy Boivin, and Jacques Corbeil. Comparison of Automated Microarray Detection with Real-Time PCR Assays for Detection of Respiratory Viruses in Specimens Obtained from Children. *Journal of Clinical Microbiology*, 47(3) :743–750, March 2009.

[239] Martin Reese, Barry Moore, Colin Batchelor, Fidel Salas, Fiona Cunningham, Gabor Marth, Lincoln Stein, Paul Flicek, Mark Yandell, and Karen Eilbeck. A standard variation file format for human genome sequences. *Genome Biology*, 11(8) :R88+, August 2010.

[240] Michael Reich, John Liefeld, Helga Thorvaldsdottir, Marco Ocana, Eliot Polk, D. K. Jang, and Jill Mesirov. GenomeSpace : An environment for frictionless bioinformatics. In *Proceedings of the 103rd Annual Meeting of the American Association for Cancer Research*, volume 72, pages 3966+. American Association for Cancer Research, April 2012.

[241] W. Resch, N. Hoffman, and R. Swanstrom. Improved success of phenotype prediction of the human immunodeficiency virus type 1 from envelope variable loop 3 sequence using neural networks. *Virology*, 288(1) :51–62, September 2001.

[242] D. D. Richman and S. A. Bozzette. The impact of the syncytium-inducing phenotype of human immunodeficiency virus on disease progression. *J. Infect. Dis.*, 169 :968–974, May 1994.

[243] Adam Roberts and Lior Pachter. Streaming fragment assignment for real-time analysis of sequencing experiments. *Nat Meth*, 10(1) :71–73, January 2013.

[244] Annie Rochette, Frédéric Raymond, Jean M. Ubeda, Martin Smith, Nadine Messier, Sébastien Boisvert, Philippe Rigault, Jacques Corbeil, Marc Ouellette, and Barbara Papadopoulou. Genome-wide gene expression profiling analysis of Leishmania major and Leishmania infantum developmental stages reveals substantial differences between the two species. *BMC Genomics*, 9 :255, 2008.

[245] M. Ronaghi, M. Uhlén, and P. Nyrén. A sequencing method based on real-time pyrophosphate. *Science*, 281(5375), July 1998.

[246] Mostafa Ronaghi. Pyrosequencing Sheds Light on DNA Sequencing. *Genome Research*, 11(1) :3–11, January 2001.

[247] Mostafa Ronaghi, Samer Karamohamed, Bertil Pettersson, Mathias Uhlén, and Pål Nyrén. Real-Time DNA Sequencing Using Detection of Pyrophosphate Release. *Analytical Biochemistry*, 242(1) :84–89, November 1996.

[248] Michael Ross, Carsten Russ, Maura Costello, Andrew Hollinger, Niall Lennon, Ryan Hegarty, Chad Nusbaum, and David Jaffe. Characterizing and measuring bias in sequence data. *Genome Biology*, 14(5) :R51+, May 2013.

[249] Jonathan M. Rothberg, Wolfgang Hinz, Todd M. Rearick, Jonathan Schultz, William Mileski, Mel Davey, John H. Leamon, Kim Johnson, Mark J. Milgrew, Matthew Edwards, Jeremy Hoon, Jan F. Simons, David Marran, Jason W. Myers, John F. Davidson, Annika Branting, John R. Nobile, Bernard P. Puc, David Light, Travis A. Clark, Martin Huber, Jeffrey T. Branciforte, Isaac B. Stoner, Simon E. Cawley, Michael Lyons, Yutao Fu, Nils Homer, Marina Sedova, Xin Miao, Brian Reed, Jeffrey Sabina, Erika Feierstein, Michelle Schorn, Mohammad Alanjary, Eileen Dimalanta, Devin Dressman, Rachel Kasinskas, Tanya Sokolsky, Jacqueline A. Fidanza, Eugeni Namsaraev, Kevin J. McKernan, Alan Williams, G. Thomas Roth, and James Bustillo. An integrated semiconductor device enabling non-optical genome sequencing. *Nature*, 475(7356) :348–352, July 2011.

[250] Jonathan M. Rothberg and John H. Leamon. The development and impact of 454 sequencing. *Nat Biotechnol*, 26(10) :1117–1124, October 2008.

[251] Joel Rousseau, Pierre Chapdelaine, Sébastien Boisvert, Luciana P. Almeida, Jacques Corbeil, Alexandre Montpetit, and Jacques P. Tremblay. Endonucleases : tools to correct the dystrophin gene. *J Gene Med*, 13(10) :522–537, October 2011.

[252] Stephen M. Rumble, Phil Lacroute, Adrian V. Dalca, Marc Fiume, Arend Sidow, and Michael Brudno. SHRiMP : Accurate Mapping of Short Color-space Reads. *PLoS Comput Biol*, 5(5) :e1000386+, May 2009.

[253] Douglas B. Rusch, Aaron L. Halpern, Granger Sutton, Karla B. Heidelberg, Shannon Williamson, Shibu Yooseph, Dongying Wu, Jonathan A. Eisen, Jeff M. Hoffman, Karin Remington, Karen Beeson, Bao Tran, Hamilton Smith, Holly Baden-Tillson, Clare Stewart, Joyce Thorpe, Jason Freeman, Cynthia Andrews-Pfannkoch, Joseph E. Venter, Kelvin Li, Saul Kravitz, John F. Heidelberg, Terry Utterback, Yu-Hui Rogers, Luisa I. Falcón, Valeria Souza, Germán Bonilla-Rosso, Luis E. Eguiarte, David M. Karl, Shubha Sathyendranath, Trevor Platt, Eldredge Bermingham, Victor Gallardo, Giselle Tamayo-Castillo, Michael R. Ferrari, Robert L. Strausberg, Kenneth Nealson, Robert Friedman, Marvin Frazier, and J. Craig Venter. The Sorcerer II Global Ocean Sampling Expedition : Northwest Atlantic through Eastern Tropical Pacific. *PLoS Biol*, 5(3) :e77+, March 2007.

[254] Kim Rutherford, Julian Parkhill, James Crook, Terry Horsnell, Peter Rice, Marie-Adèle Rajandream, and Bart Barrell. Artemis : sequence visualization and annotation. *Bioinformatics*, 16(10) :944–945, October 2000.

[255] H. Saigo, J. P. Vert, N. Ueda, and T. Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20 :1682–1689, July 2004.

[256] Jean Philippe Vert Hiroto Saigo and Tatsuya Akutsu. Optimizing amino acid substitution matrices with a local alignment kernel. *BMC Bioinformatics*, 7 :246, 2006.

[257] Yasubumi Sakakibara, Kris Popendorf, Nana Ogawa, Kiyoshi Asai, and Kengo Sato. Stem kernels for RNA sequence analyses. *J Bioinform Comput Biol*, 5(5) :1103–1122, October 2007.

[258] Kamil Salikhov, Gustavo Sacomoto, and Gregory Kucherov. Using cascading Bloom filters to improve the memory usage for de Brujin graphs. In *WABI*, 2013.

[259] Steven L. Salzberg, Adam M. Phillippy, Aleksey Zimin, Daniela Puiu, Tanja Magoc, Sergey Koren, Todd J. Treangen, Michael C. Schatz, Arthur L. Delcher, Michael Roberts, Guillaume Marçais, Mihai Pop, and James A. Yorke. GAGE : A critical evaluation of genome assemblies and assembly algorithms. *Genome research*, 22(3) :557–567, March 2012.

[260] Steven L. Salzberg and James A. Yorke. Beware of mis-assembled genomes. *Bioinformatics (Oxford, England)*, 21(24) :4320–4321, December 2005.

[261] Oliver Sander, Tobias Sing, Ingolf Sommer, Andrew J. Low, Peter K. Cheung, P. Richard Harrigan, Thomas Lengauer, and Francisco S. Domingues. Structural descriptors of gp120 V3 loop for the prediction of HIV-1 coreceptor usage. *PLoS Comput Biol*, 3(3), March 2007.

[262] F. Sanger, S. Nicklen, and A. R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12) :5463–5467, 1977.

[263] Eric E. Schadt, Michael D. Linderman, Jon Sorenson, Lawrence Lee, and Garry P. Nolan. Computational solutions to large-scale data management and analysis. *Nature Reviews Genetics*, 11(9) :647–657, September 2010.

[264] Michael Schatz. Computational thinking in the era of big data biology. *Genome Biology*, 13(11) :177+, 2012.

[265] Michael C. Schatz, Ben Langmead, and Steven L. Salzberg. Cloud computing and the DNA data race. *Nature Biotechnology*, 28(7) :691–693, July 2010.

[266] Patrick Schloss and Jo Handelsman. Metagenomics for studying unculturable microorganisms : cutting the Gordian knot. *Genome Biology*, 6(8) :229–4, August 2005.

[267] Patrick D. Schloss and Jo Handelsman. Introducing DOTUR, a Computer Program for Defining Operational Taxonomic Units and Estimating Species Richness. *Applied and Environmental Microbiology*, 71(3) :1501–1506, March 2005.

[268] Thomas Schoenfeld, Melodee Patterson, Paul M. Richardson, K. Eric Wommack, Mark Young, and David Mead. Assembly of Viral Metagenomes from Yellowstone Hot Springs. *Applied and Environmental Microbiology*, 74(13) :4164–4174, July 2008.

[269] Matthew B. Scholz, Chien-Chi Lo, and Patrick S. G. Chain. Next generation sequencing and bioinformatic bottlenecks : the current state of metagenomic data analysis. *Current Opinion in Biotechnology*, 23(1) :9–15, February 2012.

[270] Steven L. Scott and Et Al. The Cray T3E Network : Adaptive Routing in a High Performance 3D Torus. In *Proceedings of Hot Interconnects IV Symposium*, pages 147–156. Stanford University, August 1996.

[271] Nicola Segata, Levi Waldron, Annalisa Ballarini, Vagheesh Narasimhan, Olivier Jousson, and Curtis Huttenhower. Metagenomic microbial community profiling using unique clade-specific marker genes. *Nature Methods*, 9(8) :811–814, August 2012.

[272] Jay Shendure and Hanlee Ji. Next-generation DNA sequencing. *Nature Biotechnology*, 26(10) :1135–1145, October 2008.

[273] Jay Shendure, Gregory J. Porreca, Nikos B. Reppas, Xiaoxia Lin, John P. McCutcheon, Abraham M. Rosenbaum, Michael D. Wang, Kun Zhang, Robi D. Mitra, and George M. Church. Accurate multiplex polony sequencing of an evolved bacterial genome. *Science*, 309(5741) :1728–1732, September 2005.

[274] Jared T. Simpson. *Efficient sequence assembly and variant calling using compressed data structures*. PhD thesis, Queens' College, University of Cambridge, September 2012.

[275] Jared T. Simpson and Richard Durbin. Efficient de novo assembly of large genomes using compressed data structures. *Genome Research*, 22(3) :549–556, March 2012.

[276] Jared T. Simpson, Kim Wong, Shaun D. Jackman, Jacqueline E. Schein, Steven J. M. Jones, and Inanç Birol. ABySS : A parallel assembler for short read sequence data. *Genome Research*, 19(6) :1117–1123, 2009.

[277] M. Sirois, L. Robitaille, R. Sasik, J. Estaquier, J. Fortin, and J. Corbeil. R5 and X4 HIV viruses differentially modulate host gene expression in resting CD4+ T cells. *AIDS Res. Hum. Retroviruses*, 24 :485–493, March 2008.

[278] Chris S. Smillie, Mark B. Smith, Jonathan Friedman, Otto X. Cordero, Lawrence A. David, and Eric J. Alm. Ecology drives a global network of gene exchange connecting the human microbiome. *Nature*, 480(7376) :241–244, December 2011.

[279] L. M. Smith, J. Z. Sanders, R. J. Kaiser, P. Hughes, C. Dodd, C. R. Connell, C. Heiner, S. B. Kent, and L. E. Hood. Fluorescence detection in automated DNA sequence analysis. *Nature*, 321(6071) :674–679, 1986.

[280] Lincoln Stein. Creating a bioinformatics nation. *Nature*, 417(6885) :119–120, May 2002.

[281] Lincoln D. Stein. The case for cloud computing in genome informatics. *Genome Biology*, 11(5) :207+, May 2010.

[282] Jordan Stockton. BaseSpace Roadmap. In *Plant and Animal Genome XXI Conference*. Plant and Animal Genome, 2013.

[283] Andrew S. Tanenbaum. *Modern Operating Systems*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2007.

[284] J. Shawe Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[285] Washington Taylor, Jud Leonard, and Lawrence C. Stewart. Efficient tilings of de Bruijn and Kautz graphs, January 2011.

[286] Linus Torvalds. Linux : a Portable Operating System. Master's thesis, UNIVERSITY OF HELSINKI, UNIVERSITY OF HELSINKI, January 1997.

[287] Susannah G. Tringe, Christian von Mering, Arthur Kobayashi, Asaf A. Salamov, Kevin Chen, Hwai W. Chang, Mircea Podar, Jay M. Short, Eric J. Mathur, John C. Detter, Peer Bork, Philip Hugenholtz, and Edward M. Rubin. Comparative Metagenomics of Microbial Communities. *Science*, 308(5721) :554–557, April 2005.

[288] Gene W. Tyson, Jarrod Chapman, Philip Hugenholtz, Eric E. Allen, Rachna J. Ram, Paul M. Richardson, Victor V. Solovyev, Edward M. Rubin, Daniel S. Rokhsar, and Jillian F. Banfield. Community structure and metabolism through reconstruction of microbial genomes from the environment. *Nature*, 428(6978) :37–43, February 2004.

[289] Jean M. Ubeda, Danielle Légaré, Frédéric Raymond, Amin A. Ouameur, Sébastien Boisvert, Philippe Rigault, Jacques Corbeil, Michel J. Tremblay, Martin Olivier, Barbara Papadopoulou, and Marc Ouellette. Modulation of gene expression in drug resistant Leishmania is associated with gene amplification, gene deletion and chromosome aneuploidy. *Genome Biol*, 9(7), 2008.

[290] V. N. Vapnik. *Statistical learning Theory.* Wiley, New York, 1998.

[291] Thibault Varin, Connie Lovejoy, Anne D. Jungblut, Warwick F. Vincent, and Jacques Corbeil. Metagenomic profiling of Arctic microbial mat communities as nutrient scavenging and recycling systems. *Limnology and Oceanography*, 55(5) :1901–1911, 2010.

[292] Thibault Varin, Connie Lovejoy, Anne D. Jungblut, Warwick F. Vincent, and Jacques Corbeil. Metagenomic Analysis of Stress Genes in Microbial Mat Communities from Antarctica and the High Arctic. *Applied and Environmental Microbiology*, 78(2) :549–559, January 2012.

[293] Bala M. Venkatesan and Rashid Bashir. Nanopore sensors for nucleic acid analysis. *Nature Nanotechnology*, 6(10) :615–624, September 2011.

[294] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, J. D. Gocayne, P. Amanatides, R. M. Ballew, D. H. Huson, J. R. Wortman, Q. Zhang, C. D. Kodira, X. H. Zheng, L. Chen, M. Skupski, G. Subramanian, P. D. Thomas, J. Zhang, Gabor L. G. Miklos, C. Nelson, S. Broder, A. G. Clark, J. Nadeau, V. A. McKusick, N. Zinder, A. J. Levine, R. J. Roberts, M. Simon, C. Slayman, M. Hunkapiller, R. Bolanos, A. Delcher, I. Dew, D. Fasulo, M. Flanigan, L. Florea, A. Halpern, S. Hannenhalli, S. Kravitz, S. Levy, C. Mobarry, K. Reinert, K. Remington, J. Abu Threideh, E. Beasley, K. Biddick, V. Bonazzi, R. Brandon, M. Cargill, I. Chandramouliswaran, R. Charlab, K. Chaturvedi, Z. Deng, V. Di Francesco, P. Dunn, K. Eilbeck, C. Evangelista, A. E. Gabrielian, W. Gan, W. Ge, F. Gong, Z. Gu, P. Guan, T. J. Heiman, M. E. Higgins, R. R. Ji, Z. Ke, K. A. Ketchum, Z. Lai, Y. Lei, Z. Li, J. Li, Y. Liang, X. Lin, F. Lu, G. V. Merkulov, N. Milshina, H. M. Moore, A. K. Naik, V. A. Narayan, B. Neelam, D. Nusskern, D. B. Rusch, S. Salzberg, W. Shao, B. Shue, J. Sun, Z. Wang, A. Wang, X. Wang, J. Wang, M. Wei, R. Wides, C. Xiao, C. Yan, A. Yao, J. Ye, M. Zhan, W. Zhang, H. Zhang, Q. Zhao, L. Zheng, F. Zhong, W. Zhong, S. Zhu, S. Zhao, D. Gilbert, S. Baumhueter, G. Spier, C. Carter, A. Cravchik, T. Woodage, F. Ali, H. An, A. Awe, D. Baldwin, H. Baden, M. Barnstead, I. Barrow, K. Beeson, D. Busam, A. Carver, A. Center, M. L. Cheng, L. Curry, S. Danaher, L. Davenport, R. Desilets, S. Dietz, K. Dodson, L. Doup, S. Ferriera, N. Garg, A. Gluecksmann, B. Hart, J. Haynes, C. Haynes, C. Heiner, S. Hladun, D. Hostin, J. Houck, T. Howland, C. Ibegwam, J. Johnson, F. Kalush, L. Kline, S. Koduru, A. Love, F. Mann, D. May, S. McCawley, T. McIntosh, I. McMullen, M. Moy, L. Moy, B. Murphy, K. Nelson, C. Pfannkoch, E. Pratts, V. Puri, H. Qureshi, M. Reardon, R. Rodriguez, Y. H. Rogers, D. Romblad, B. Ruhfel, R. Scott, C. Sitter, M. Smallwood, E. Stewart, R. Strong, E. Suh, R. Thomas, N. N. Tint, S. Tse, C. Vech, G. Wang, J. Wetter, S. Williams, M. Williams, S. Windsor, E. Winn Deen, K. Wolfe, J. Zaveri,

K. Zaveri, J. F. Abril, R. Guigó, M. J. Campbell, K. V. Sjolander, B. Karlak, A. Kejariwal, H. Mi, B. Lazareva, T. Hatton, A. Narechania, K. Diemer, A. Muruganujan, N. Guo, S. Sato, V. Bafna, S. Istrail, R. Lippert, R. Schwartz, B. Walenz, S. Yooseph, D. Allen, A. Basu, J. Baxendale, L. Blick, M. Caminha, J. Carnes Stine, P. Caulk, Y. H. Chiang, M. Coyne, C. Dahlke, A. Mays, M. Dombroski, M. Donnelly, D. Ely, S. Esparham, C. Fosler, H. Gire, S. Glanowski, K. Glasser, A. Glodek, M. Gorokhov, K. Graham, B. Gropman, M. Harris, J. Heil, S. Henderson, J. Hoover, D. Jennings, C. Jordan, J. Jordan, J. Kasha, L. Kagan, C. Kraft, A. Levitsky, M. Lewis, X. Liu, J. Lopez, D. Ma, W. Majoros, J. McDaniel, S. Murphy, M. Newman, T. Nguyen, N. Nguyen, M. Nodell, S. Pan, J. Peck, M. Peterson, W. Rowe, R. Sanders, J. Scott, M. Simpson, T. Smith, A. Sprague, T. Stockwell, R. Turner, E. Venter, M. Wang, M. Wen, D. Wu, M. Wu, A. Xia, A. Zandieh, and X. Zhu. The sequence of the human genome. *Science*, 291(5507) :1304–1351, February 2001.

[295] Francesco Vezzi, Giuseppe Narzisi, and Bud Mishra. Reevaluating Assembly Evaluations with Feature Response Curves : GAGE and Assemblathons. *PLoS ONE*, 7(12) :e52210+, December 2012.

[296] René L. Warren, Granger G. Sutton, Steven J. M. Jones, and Robert A. Holt. Assembling millions of short DNA sequences using SSAKE. *Bioinformatics*, 23(4) :500–501, February 2007.

[297] J. D. Watson and F. H. Crick. Molecular structure of nucleic acids ; a structure for deoxyribose nucleic acid. *Nature*, 171(4356) :737–738, April 1953.

[298] David Weese, Anne K. Emde, Tobias Rausch, Andreas Döring, and Knut Reinert. RazerS–fast read mapping with sensitivity control. *Genome Res*, 19(9) :1646–1654, September 2009.

[299] David Weese, Manuel Holtgrewe, and Knut Reinert. RazerS 3 : Faster, fully sensitive read mapping. *Bioinformatics*, 28(20) :2592–2599, August 2012.

[300] Barbara Wold and Richard M. Myers. Sequence census methods for functional genomics. *Nature Methods*, 5(1) :19–21, January 2008.

[301] K. Eric Wommack, Jaysheel Bhavsar, and Jacques Ravel. Metagenomics : Read Length Matters. *Applied and Environmental Microbiology*, 74(5) :1453–1463, March 2008.

[302] Dongying Wu, Philip Hugenholtz, Konstantinos Mavromatis, Rudiger Pukall, Eileen Dalin, Natalia N. Ivanova, Victor Kunin, Lynne Goodwin, Martin Wu, Brian J. Tindall, Sean D. Hooper, Amrita Pati, Athanasios Lykidis, Stefan Spring, Iain J. Anderson, Patrik D/'haeseleer, Adam Zemla, Mitchell Singer, Alla Lapidus, Matt Nolan, Alex Copeland, Cliff Han, Feng Chen, Jan-Fang Cheng, Susan Lucas, Cheryl Kerfeld, Elke Lang,

Sabine Gronow, Patrick Chain, David Bruce, Edward M. Rubin, Nikos C. Kyrpides, Hans-Peter Klenk, and Jonathan A. Eisen. A phylogeny-driven genomic encyclopaedia of Bacteria and Archaea. *Nature*, 462(7276) :1056–1060, December 2009.

[303] Gary D. Wu, Jun Chen, Christian Hoffmann, Kyle Bittinger, Ying-Yu Y. Chen, Sue A. Keilbaugh, Meenakshi Bewtra, Dan Knights, William A. Walters, Rob Knight, Rohini Sinha, Erin Gilroy, Kernika Gupta, Robert Baldassano, Lisa Nessel, Hongzhe Li, Frederic D. Bushman, and James D. Lewis. Linking long-term dietary patterns with gut microbial enterotypes. *Science (New York, N.Y.)*, 334(6052) :105–108, October 2011.

[304] Shungao Xu, Xinxiang Huang, Huaxi Xu, and Chiyu Zhang. Improved prediction of coreceptor usage and phenotype of HIV-1 based on combined features of V3 loop sequence using random forest. *J Microbiol*, 45(5) :441–446, October 2007.

[305] Nazar M. Zaki, Safaai Deris, and Rosli Illias. Application of string kernels in protein sequence classification. *Applied bioinformatics*, 4(1) :45–52, 2005.

[306] Daniel R. Zerbino. *Genome assembly and comparison using de Bruijn graphs*. PhD thesis, University of Cambridge, September 2009.

[307] Daniel R. Zerbino and Ewan Birney. Velvet : algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 18(5) :821–829, May 2008.

[308] Daniel R. Zerbino, Gayle K. McEwen, Elliott H. Margulies, and Ewan Birney. Pebble and Rock Band : Heuristic Resolution of Repeats and Scaffolding in the Velvet Short-Read de Novo Assembler. *PLoS ONE*, 4(12) :e8407+, December 2009.

[309] L. Q. Zhang, P. Robertson, E. C. Holmes, A. Cleland, A. J. Leigh Brown, and P. Simmonds. Selection for specific V3 sequences on transmission of human immunodeficiency virus. *J. Virol.*, 67 :3345–56, 1993.

[310] Zhengwei Zhu, Beifang Niu, Jing Chen, Sitao Wu, Shulei Sun, and Weizhong Li. MGAviewer : a desktop visualization tool for analysis of metagenomics alignment data. *Bioinformatics (Oxford, England)*, 29(1) :122–123, January 2013.