Chapitre 1

TODO

— Assemblage : Etat de l'art
— Correction : Continuer état de l'art + trier par idée
— Lire Hybrid SPAdes

Chapitre 2

Année 1 : 2016 - 2017

2.1 Jour 1

— Relecture mémoire M2
— Relecture résultats
— Réappropriation sujet

2.2 Jour 2

— Correction de bugs dans l'algo de correction
— Test de l'algo

2.3 Jour 3

— Lecture article CoLoRMap
— Renommage des SR test par leur nom correct
— Tests et étude des overlaps

2.4 Jour 4

— Etude des propriétés des autres outils de correction
— Appropriation des idées pour le notre
— Remplir les gaps avec les bases non corrigées du RL (idée + début)

2.5 Jour 5

— Papiers administration
— Achat voiture

2.6 Jour 8

— Possibilité de remplir les gaps avec bases non corrigés du RL si gap < maxgap
— RL ainsi produits : Bases corrigées en min, bases erronées en maj
— Début automatisation traitement en C + débarrassage fichiers seeds

2.7 Jour 9

— Fin automatisation traitement en C
— Tests comparatifs outils automatisé / outil géré par bash
— => Automatisation plus lente que de passer par bash
— Test de la qualité des résultats f(maxgap)

2.8 Jour 10

— Automatisation création du fichiers de seeds (en bash)
— Automatisation des tests (run algo, alignement résultats, calcul précision)
— Installation PgSA
— Dénombrement des templates bien couverts (maxgap = 0)

2.9 Jour 11

— Séminaire (P. Caron)
— Papiers administratifs (création agent / financement machine)
— Dénombrement des templates bien couverts
— Premier test sur PgSA

2.10 Jour 12

— Dénombrement des templates bien couverts

— Pblat : lmin rendu paramétrable

— Pblat : seuil pour match total (aligné sur >= xx% de la longueur = match total) rendu paramétrable

2.11 Jour 15

— Pblat : Pas de lmin inutiles dans les fonctions (visiblement)

— Algo : Plus de snprintf, remplacés par des fprintf

— Algo : get_tpl_seq corrigé -> Ne prend le tpl que si pas suivi d'un autre chiffre

— Algo : Premier clean-up de code + optimisation

— Algo : Tests en variant lmin / lmax

— Devis PC

2.12 Jour 16

— Tests sur lmin / lmax / th et conclusions

— Comparatifs pblat basique / pblat modifié

— Début lecture survey

— Commentaires .h

— Algo : Besoin d'un consensus entre les seeds

— Déménagement frigo (1h)

2.13 Jour 17

— Récupération jeu de données bien nommé et complet

— Correction script stats, pour l'adapter au nouveau jeu de données + modification recherche longueur du template (virer le _1 à la fin du nom du contig)

— Recalcul des alignements SR sur LR (fichiers psl)

— Recalcul des résultats (SR 1 & 2 et params par défaut) + tableau

— 2ème étape de recrutement semble inutile (peu de pref / suff recrutés)

— => Vérification si préfixes / suffixes présents dans les gaps des tpl corrigés

— Survol du survey de correction, semble peu intéressant, bien que présente les différents types d'erreurs en fonction de la technologie de séquençage choisie, les fréquences / seuils / tailles pour les k-mers, subs vs subs + indels, modèles statistiques pour les erreurs, les répétitions / haplotypes, méthodes de corrections en fonction de plateforme utilisée => RIEN SUR NANOPORE A PART NANOCORR

— Installation de NaS et premiers tests

— Ebauche poster fête de la science

2.14 Jour 18

— Recalcul des résultats (SR 1 & 2 et params par défaut) + tableau

— Dénombrement des pref / suff recrutés lors de la deuxième étape (lmax = 10 / lmax =50)

— => Extrêmement peu de recrutement, aussi bien dans 1D que 2D

2.15 Jour 19

— Recalcul des résultats (SR 1 & 2 et params par défaut)

— Réunion w/ TL

— Installation et problèmes avec NaS

— Gros problèmes de Wi-Fi

— Signature contrat + Documents mission / remboursement / etc

2.16 Jour 20

— Recalcul des résultats (SR 1 & 2)

— Noms uniques pour fichier seeds et tmp.psl

— Possibilité de lancer plusieurs instances de l'algo en même temps

— Travail sur poster

2.17 Jour 21

— Recalcul des résultats (SR 1 & 2)

— MinION6 2D : EXTREMEMENT LONG

— Travail sur poster (Intro / schéma / comparatif séquenceurs) -> une moitié terminée

2.18 Jour 22

— Poster terminé (8h -> 13h30)

— Tableau résultats SR1 & SR2

— Tableau résultats filtrés (qualité de tous les contigs > 90% pour pouvoir corriger un read)

— NaS => Problème = Newbler (runAssembly project)

2.19 Jour 23

— Tableau résultats filtrés (un contig avec q > 90% pour pouvoir corriger un read)

— Création de deux scripts de filtrage (all contigs >= 90 & un contig >= 90) + génération auto ligne tableau résultats

— Tableau comparatif NaS / nous

— Tests sur NaS, toujours pas fonctionnel => path à modifier pour last / runassembly

— Résumé Colormap sur pdf

— Réunion w/ TL & AL, pistes de travail (mates, utilisation de GkA, assemblage SR puis mapping LR)

2.20 Jour 24

— Installation de Minia + premiers tests assemblage & mapping RL sur SR assemblés + tableau résultats + stats rapides des LR matchant sur plusieurs contigs

— Début lecture papier Minia (reste 1 page)

— Installation de CoLoRMap + test perfs => très long

— Recherche pblat paired end => Ne semble pas être possible

— Installation inchworm (pour pblat en paired end) => Bug avec psl2sam.pl

— Résultats algo sans 2nde étape (calculs)
— Erreur MinION : Détection par 6 nucléotides, donc séparation difficiles si homopolymères de longueur > 6 (suppressions favorisées, 66% des erreurs observées)
— Commande pour laisser programmes tourner sur le serveur : nohup

2.21 Jour 25

— Fin lecture Minia
— Séminaire Lyndon
— Fin temps exec algo (étape 1 seulement) avec SR1 & SR2 => Fail car .fa résultat non supprimé lors de l'exécution => Relancement des tests en soirée
— Test Colormap sur jeu exemple du papier + sur ensemble de LR MinION => Beaucoup trop long sur MinION (+ de 3h)
— Algo : Suppression des fichiers temporaires une fois le traitement terminé
— Revue poster avec mme Selmi

2.22 Jour 26

— Modifications poster
— Tableau résultats (Etape 1 seulement, SR1 & SR2 concaténés)
— Récupération temps exec Colormap sur MinION : + de 16h => TROP LONG
— Recherche sur inchworm, pour maper avec BLAT en paired-end
— Calcul résultats avec Etape 1 ET Etape 2 (SR1 & SR2 concaténés)

2.23 Mois 2 : Octobre 2016

2.24 Jour 29

— Visite médicale
— Parallel : Possibilité de lancer l'algo sur x cores en même temps => Enorme gain de temps
— Paired-end reads : Impossible d'estimer la distance entre les deux sans mapping
— Nouveaux tests sur Minia => Non déterministe, ne produit jamais le même assemblage

2.25 Jour 30

— Remise au propre du diary

— Recalcul des temps d'exécution avec parallel

— Tableau assemblage puis mapping LR sur PDF pour réunion

— Statistiques sur pairs de reads mappés sur même LR template

— Statistiques sur distance entre les deux reads d'une paire

— Réunion w/ TL & AL

2.26 Jour 31

— Tri des articles dans répertoires

— Lecture BLAST

— Comparaison alignement PBLAT / BLAT / BLAST sur SR assemblés => Production fichier BLAST avec tous les alignements

— Filtrage des alignements (lmin <= |al| <= lmax) => Production fichier temporaire avec les alignements significatifs

— Filtrage des alignements pour trouver les LR mappés sur différents contigs => Production fichier temporaire avec les contigs liable

— Dénombrement et calcul taille moyenne des contigs liables / non liables par les LR

2.27 Jour 32

— Analyse du fichier de contigs liables, afin de déterminer les liens et gaps entre les contigs liés par un même LR (matin)

— Filtrage des gaps / overlaps trop longs entre contigs sur lesquels s'est mappé un même LR (début) => comptage des contigs liables obtenus (après midi)

— Tentative automatisation du filtrage des alignements, du calcul des LR alignés sur plusieurs contigs et du filtrage des gaps trop longs avec parallel => GROS FAIL, GROSSE PERTE DE TEMPS, GROSSE GALERE A RETROUVER LES BONS RESULTATS (fin après midi)

2.28 Jour 33

— Filtrage des gaps / overlaps trop longs entre contigs sur lesquels s'est mappé un même LR => Production du fichier final permettant de générer le graphe

— Reprogrammation de filterAlignments en C (getMultimaps et filterGaps bash plus rapide car utilisation de grep dans le script) => filterAlignments maintenant instantané

— Automatisation de la création du fichier final permettant de générer le graphe (Temps total : 2 min avec les 6 jeux MinION (All SR) - 2 min avec les 6 jeux MinION (|250| SR))

— Correction de bugs dans le script générant le fichier final permettant de générer le graphe (car production de mauvais résultats en cas d'enchainement de contigs non liable / liable / non liable / liable)

— Analyse d'un graphe minimal, afin de voir comment implémenter => Graphé orienté, noeuds = contigs, edges = LR + |gap|

2.29 Jour 36

— Programmation d'une structure de graphe, afin de construire le graphe d'assemblage

— Modification du format du fichier permettant de générer le graphe

— Parcours du fichier de graphe et génération du graphe

— Programmation d'une fonction de parcours du graphe explorant tous les successeurs possible de chaque noeud

— Etude des résultats obtenus

2.30 Jour 37

— Re-génération du graphe avec des différents paramètres => Semble plus cohérent de fixer un gap max et un overlap max plutôt qu'un gap et une extension min

— Reprise de la fonction de parcours afin de réaliser la plus grande extension possible lors de l'exploration de chaque noeud

— Etude des résultats obtenus (longueur max d'un contig VS longueur de l'assemblage), et de leur pertinence

— Plus long contig produit par Minia jamais liable => Nécessité d'un seuil d'overlap / gap max relatif à la longueur du contig et du LR ?

— L'assemblage en cas d'overlap semble produire des contigs de mauvaise qualité

— Reprogrammation de filterGaps en C

2.31 Jour 38

— Modification du filtrage des multimaps : Au lieu de chercher les LR mappés sur plusieurs contigs, on cherche juste les LR avec plusieurs hits

— Reprogrammation du fitlrage des multimaps en C : Maintenant instantané

— Filtrage des multimaps inutile : Le filtrage des gaps écarte déjà les LR avec un unique mapping, et vérifie, en cas de double mapping, que les deux contigs de référence sont bien différents

— => Génération du fichier du graphe maintenant uniquement en C (sauf pour le lancement / tris de fichier : bash) et instantanée

— Ajout des free dans tous les programmes en C (Multimaps / filterGaps / assembleContigs)

— Génération de graphes avec différents maxGap pour les deux sous ensembles de LR pour réunion

— Nécessité d'avoir la position de début du gap en plus de sa longueur dans les edges du graphe

2.32 Jour 39

— Fête de la science

— Réunion w/ TL

2.33 Jour 40

— Fête de la science

2.34 Jour 43

— Correction mineure dans algo correction MinION

— Génération des scaffolds lors du parcours du graphe

— Dénombrement couples / scaffolds en fonction de |al| min alignement => Tableau

— Stats. sur les scaffolds produits (longueur, qualité, % du génome couvert, ...) en fonction de |al| min, |al| max = 500, plusieurs passages par noeud

2.35 Jour 44

— Stats. sur les scaffolds produits (longueur, qualité, % du génome couvert, ...) en fonction de |al| min, |al| max = 500, plusieurs passages par noeud

— Modification de la génération des scaffolds afin de pouvoir autoriser un petit overlap

— Stats sur les scaffolds GAP + OVERLAP + plusieurs passages => Semble peu intéressant car perte rapide de qualité, influence variable sur avLen des scaffolds + Minia aurait trouvé les overlaps

— Stats sur les scaffolds produits en fonction de |al| min, |al| max = 500, en ne passant qu'une fois par chaque noeud du graphe => Semble plus prometteur, bons résultats

— Augmentation de |al| max -> Autant de "couples", mais plus de scaffolds, génome de référence mieux couvert => Plus intéressant

2.36 Jour 45

— Stats sur les scaffolds GAP + OVERLAP + un passage => Semble peu intéressant car perte rapide de qualité, influence variable sur avLen des scaffolds + Minia aurait trouvé les overlaps

— Recherche du seuil |al| min et |al| max idéal pour obtenir des scaffolds de bonne qualité et bien couvrir le génome de référence

— Mise en évidence d'un problème dans le parcours : Parfois impossible de "rabouter" plusieurs scaffolds qui devraient normalement n'en former qu'un (+ dessin dans PDF réunion)

— Stats sur les scaffolds produits en fonction de |al| min, |al| max = 500, en ne passant qu'une fois par chaque noeud du graphe et en raboutant les contigs => Mieux que de pouvoir repasser infiniment par chaque noeud, mais moins bon que de n'y passer qu'une fois

— Statistiques et comparaisons des différentes méthodes de parcours du graphe (avec et sans raboutage), avec variation maxGap, maxOverlap, ...

2.37 Jour 46

— Gestion de mode de production de scaffolds raboutés : Toutes les parties + le tout OU juste le tout

— Tout ce qui a été fait précédemment était faux, car les contigs sans successeur étaient reportés dans le fichier de résultat => reprise de TOUS les tableaux

— Réunion w/ TL, quelques pistes, mais peu d'idées

2.38 Jour 47

— Relecture des résultats de notre GA => Parcours sans marquer les noeuds et avec variations sur |al| peut potentiellement mieux couvrir => Tableau => bonne couverture mais mauvaise qualité

— Choisir d'aller vers le noeud le plus proche à chaque fois permet d'augmenter la qualité mais diminue la couverture

— Mise en évidence d'un problème avec le raboutage => Produit un scaffold trop court => Correction à l'aide d'un MWE (on copiait le contig au lieu du scaffold)

— Nouveaux tests avec le raboutage corrigé => Permet de BEAUCOUP MIEUX couvrir mais perte de qualité

— L'exécution sur l'ensemble de reads Illumina de taille 250 semble produire des résultats similaires

— Lecture du papier de Karlsson, totalement inintéressant => Détails sur reads MinION et sur le fait qu'on peut scaffolder avec + paramètres utilisés pour blast => beaucoup trop long (10-12h)

— Nouveaux tests, overlaps intéressants => Améliore avLEn, avQual, et genCov => très proche de la couverture à 100% (idéal à 2100 -150 1, si on ne va QUE vers le noeud le plus long à chaque fois, sans poursuivre avec un autre si déjà visité)

— Reprise de tous les tableaux, car lors de l'exploration d'un noeud, si le noeud permettant l'extension max est déjà visité, on peut poursuivre maintenant poursuivre avec un autre noeud => Permet de mieux couvrir, qualité similaire

2.39 Jour 48

— Fin du remplissage des tableaux

— Nouvelle idée de parcours : Tout parcourir même si déjà visité, mais ne pas sortir les scaffolds commençant à un noeud se trouvant en milieu de parcours => Longueur et couverture similaire, meilleure qualité, mais TRÈS LONG

2.40 Jour 50

— Assemblage avec ABYSS => Moins de contigs, visiblement de meilleure qualité et longueur, mais beaucoup plus long

— Mise en évidence d'un bug dans assembleContigs, lors de la lecture du fichier du graphe le contig destination est copié avec le
n final => Correction

— Tests de GA avec les contigs générés par ABYSS => Permet de quasiment tout couvrir mais assez mauvaise qualité pour le peu de bases LR introduites

— Tests de GA avec les contigs Minia, en sortant les contigs non traversés => Augmente la qualité et couvre un peu plus

— Stats sur les distances moyennes entre deux contigs liés par notre algo en fonction de de |al| min => contigs liés généralement trop loin l'un de l'autre pour obtenir une bonne qualité et couverture

2.41 Jour 51

— Début lecture Canu

— GA avec filtration des LR (on ne garde que les LR, |LR| <= 10k) => Peu concluant

— Idem avec les reads 2D seulement => Peu concluant

— Tentative de réalignement avec BLAST des LR sur les scaffodls obtenus + réexécution de l'algo => Très mauvais résultats, autant en qualité qu'en couverture

— Tentative de polishing des résultats de la GA avec Pilon => Permet un gain de qualité et un gain de couverture => Intéressant de run plusieurs fois ?

2.42 Jour 52

— Fin lecture Canu

— Suppression des fichiers inutiles du serveur

2.43 Jour 53

— Tentatives de polishing, mais bugs avec Pilon => Bizarre car marchait la veille

— Réécriture du fonctionnement de Canu sur PDF + éclaircissement de certains points

— Calcul des distances de mappings entre scaffold produit et contig le plus à gauche du scaffold => Généralement très éloignés l'un de l'autre

— Réunion w/ TL & TL (27/10/16) => Nouvelle pistes de travail (Consensus sur les LR dans les gaps, nouveau jeu de données, ...)

2.44 Jour 54

— Téléchargement nouveau jeu de données (Ecoli)

— Tri et extraction de données du grand ensemble de SR

— Assemblage des SR avec Minia + stats

— Dénombrement et étude des LR différents permettant de lier 2 contigs (avec variation |al| min max et gap min max, sur ADP) => Consensus possible => On fait le consensus de toutes les portions de LR permettant de combler le gap entre 2 contigs en fixant la taille de chacune de ces portions à la taille du plus grand gap

— Test Canu sur ADP1 => 6h pour run

— EBI down : Impossible de télécharger les MinION de Ecoli

2.45 Jour 57

— Étude des résultats de Canu sur ADP1 => Galère avec BWA

— Téléchargement des fichiers MinION de Ecoli (MAP005) + test de notre GA => Produit de très mauvais résultats => Les LR de MAP005 sont pourris, alignements à < 1% d'identité sur génome de référence

— Test Canu sur MAP005 => Ne fonctionne pas => MAP005 trop pourri

— Début génération consensus des LR permettant de lier 2 contigs

2.46 Jour 58

— Re-subsampling des SR Ecoli (pour Abyss)

— Assemblage des SR Ecoli avec Abyss et Minia + stats

— Génération consensus des LR permettant de lier 2 contigs

— Test GA avec consensus sur ADP1 => Bug

— Téléchargement MinION Ecoli (MAP006) => Meilleur que MAP005, alignements à 70% identité

— Test Canu sur MAP006

2.47 Jour 59

— Analyse résultats Canu (MAP006 et ADP1) => Excellents résultats

— Correction bug génération consensus => On dépassait parfois la fin de certains LR, d'où des résultats non déterminés

— Test GA sur ADP1 ; avec consensus (Minia et Abyss)

— Test GA sur MAP006 ; assemblage Minia ; sans consensus

— Test GA sur MAP006 ; assemblage Minia ; avec consensus

— Comparaison résultats avec / sans consensus => Consensus semble inutile car implique baisse de qualité et de couverture

— Nouvelle idée : Générer les "contigs" produits par méthode stage en corrigeant les LR, les blaster sur les contigs, et réaliser la GA avec ces résultats plutôt avec les LR

— Nouvelle idée : Corriger les LR avec Canu, blaster les LR corrigés sur les contigs, et réaliser la GA avec ces résultats plutôt qu'avec les LR non traités

2.48 Jour 60

— Tests et étude des résultats de la GA avec LR corrigés par Canu / parties "corrigées" par méthode du stage

— Bug dans la production de scaffolds :
n parfois copié => Car reads corrigés par canu n'avaient pas un nom unique => corrigé

— Nouvelle idée : Générer les "contigs" produit par méthode stage en corrigeant les LR, et les assembler avec Minia / Abyss => Prometteur !

— => Mais PBLAT des SR sur les LR Ecoli extrêmement long

— Nouvelle idée : Tester l'alignement des séquences produites avec dnadiff => Intéressant seulement si très peu de contigs très longs

— Description des jeux de données (SR Ecoli, "contigs" produits par méthode stage, LR corrigés par Canu) sur PDF Réunions

— Réunion with AL & TL (03/11/16) => Nouvelle idée prometteuse, mais nécessité de comparer les k-mers SR / LR

— Blast des SR sur les LR => Bcp trop long

— PBLAT de 1M SR sur LR Ecoli => 34 min, mais exécution de l'algo beaucoup trop lente

2.49 Jour 61

— Installation et tests de Bowtie2

— Test de perfs des aligneurs (sur 1M et 12M reads Ecoli)

— Ajout contigs SR et contigs régions correctes de LR dans un fichier puis assemblage => Peu intéressant

— Ajout contigs SR et régions correctes de LR dans un fichier puis assemblage => Peu intéressant

— Assemblage des régions correctes de LR avec Abyss => Possible avec ABYSS et non abyss-pe, mais produit de mauvais résultats qq soit taille kmer

— Comparaison Abyss / Abyss 2.0.2 en paired-ends => 2.0.2 plus efficace, produit moins de contigs, globalement plus longs => Mais une fois contigs courts filtrés, résultats similaires, 2.0.2 couvre 0,2 % du génome en plus

— Programmation script bash pour récupérer SR non mappés

— Ajout SR inutilisés et régions correctes de LR dans un fichier puis assemblage => Semble peu intéressant, quelle que soit la combinaison utilisée, bien que test approximatif, car SR mal nommés lors du mapping (non prise en compte des /1 /2)

— Etude format sortie Bowtie2 (SAM) pour voir comment récupérer les régions correctes des LR

— Lecture Jabba => Très intéressant

2.50 Jour 63

— Tests récupération parties correctes LR Ecoli, avec algo Stage, 1M SR alignés avec PBLAT => Extrêmement long

— Tests perf Bowtie vs Bowtie2 vs SOAP2

— Génération parties correctes des LR à partir de fichier SAM => Régions des LR extraites de bonne qualité

— Test assemblage parties correctes LR Ecoli, avec algo Filtration, 1M SR alignés avec Bowtie2, very-fast => Mauvais résultats GA

— Test assemblage parties correctes LR Ecoli, avec algo Filtration, 12M SR p-e alignés avec Bowtie2, very-fast => Mauvais résultats GA

— Recalcul des alignements SR sur LR avec PBLAT, pour ADP1, avec SR renommés (prise en compte des /1 /2)

2.51 Jour 64

— Retravail sur script récupérant les SR non mappés => Maintenant avec parallel, mais toujours long

— Ajout des SR complets et régions correctes / contigs LR dans un même fichier puis assemblage => Pas concluant (pour ADP1)

— Test assemblage parties correctes LR Ecoli, avec algo Filtration, 12M SR s-e alignés avec Bowtie2, very-fast => Mauvais résultats GA

— Test assemblage parties correctes LR ADP1, avec algo Filtration, 12M SR s-e alignés avec Bowtie2, very fast => Mauvais résultats GA

— => Assembler les parties correctes des LR ne sert donc à rien

— Test de Guided Assembly sur Ecoli, avec Abyss => Aussi peu satisfaisant que Minia

— Début de tentative d'ordre des contigs SR avec les LR. => Construction d'un graphe pour "parcourir" les contigs liés => Poids d'une "arrête" = nombre de LR permettant de lier c2 à droite de c1

— Lecture spaced k-mer machin => Peu intéressant, orienté résultats

— Nouvelle idée : Ajouter les reverse-complement de chaque contig => Plus mauvais résultats que sans RC pour la GA, mais semble intéressant pour l'ordre

— Nouvelle idée : Concaténer tous les LR et mapper les contigs SR sur le résultat obtenu => BWA mem ne passe pas

2.52 Jour 65

— Programmation d'un graphe et d'algos de parcours permettant de générer l'ordre des contigs

— Filtrer les alignements en fonction de leur identité => Améliore les résultats de la GA, mais pas suffisamment

— Ordonner les contigs avec les LR semble difficile, même en utilisant les RC des contigs

— Nouvelle idée : Ajouter les RC des LR également => Mauvais résultats pour la GA

— Ordre des contigs avec RC contigs et RC LR => Semble possible d'en tirer qq chose

2.53 Jour 66

— Ajout SR inutilisés et régions correctes / contigs LR / etc dans un même fichier puis assemblage, avec l'intégralité des SR non mappés, pour ADP1 avec Algo correction => Pas concluant

— Lecture de SSPACE-Longread

— Test de SSPACE-Longread => Permet effectivement de relier certains contigs

— Ordonner les contigs avec mapping des LR dessus est donc possible

— Nouvelle idée : Poids d'une arrête = Nombre de LR permettant de lier c2 à droite de c1 + qualité des alignements => Parcours du graphe permettant de relier les contigs en se basant sur la qualité des alignements prometteur

— Nouvelle idée : Poids d'une arrête = Nombre de LR permettant de lier + qualité * longueur des alignements = nombre de bases correctes alignées => Essayer de déterminer les liens avec le nombre moyen de bases correctement alignées par lien (qual * len / nbLR)

— Réunion w/ TL (10/11/16) => Continuer à essayer d'ordonner les contigs, voir comment comparer les k-mers (diagramme de Venn)

— Nouvelle idée : Filtrer les alignements pour ne conserver les liens entre deux contigs que s'ils sont préfixes / suffixes, et avec un alMin de taille la longueur des k-mers => Toujours pas concluant pour la GA, même en réduisant gapMax, la distance réelle entre deux contigs joints reste trop grande, mais semble prometteur pour l'ordre

— Combiner poids des arrêtes = (qual * len / nbLR) + liens pref / suff uniquement avec alMin = k => Semble prometteur, mais toujours certains contigs mal ordonnés => nécessité de trouver pourquoi

2.54 Jour 67

— Prendre en compte si les LR mappés sont RC ou non
— Positions début / fin de mapping des LR => Apporte de l'info ? => Oui, end < beg signifie que le hit est en resverse complement
— Prendre en compte alEnd > alBeg pour les LR et ne prendre un compte que les alignements préfixes / suffixes sur les contigs => Pas concluant pour la GA (meilleure qualité mais bien moins bonne couverture), permet de bien ordonner les contigs, mais un contig n'est généralement lié qu'à un seul autre, qui est parfois trop loin => pas assez de liens pour conclure qqch
— Régions qui se mappent plusieurs fois = régions répétées ? => Visiblement non
— Les overlaps ont tendance à introduire des erreurs dans les placements

2.55 Jour 70

— Compareads => Compare les reads et non les kmers
— Problème : nb de bases correctes = faux, car on ne prend en compte les alignements que d'un seul côté du lien, donc pas possibilité d'en tirer une info pertinente
— Conséquence : Poids des arrêtes = nombre de LR reliant c1 à c2 et c'est tout
— Compter l'intégralité des liens entre 2 contigs et ordonner en prenant en compte ce poids (ie si c1 c1 c1 c3 liés par un LR, on aura un poids de 3 pour c1 -> c3)
— Nécessité : Trouver les reads qui se mappent à plusieurs endroits et les splitter
— Ajout des RC des LR et des contigs => Plus de mappings, mais que des doublons
— Vérifier que end > beg pour ne pas avoir du duplicata avec les RC des reads et/ou des contigs (car end > beg = Mapping sur RC) => Sans vérification, tous les alignements sont produits en double
— Compter le nombre de mapping sur contig / RC(contig) afin de déterminer lequel est correct, avec résultats BLAST => En comptant le nombre de mapping à l'endroit / à l'envers de chaque contig dans le fichier blast : Pas concluant aussi bien pour ADP que Ecoli => En comptant le nombre de contigs liés à gauche et à droite dans le graphe : Pas concluant aussi bien pour ADP que Ecoli
— Réunion ED
— Meilleurs résultats avec un autre aligneur ? => Téléchargement de BLASR, comme dans SSPACE-Longread, car fait pour aligner des LR
— Ajouter les RC des LR et des contigs est inutile, car BLAST et BLASR mappent quer -> ref dans les deux sens

2.56 Jour 71

— Test de scaffolding avec les résultats de BLASR => Aussi peu intéressant qu'avec BLAST

— Comparaison résultats SSPACE / nous afin de voir où ça bloque => Probablement filtration

— Récupération du filtre de filtration de SSPACE, et production des alignements conservés dans un fichier => Produit un graphe permettant de lier correctement les contigs

— Comme pour SSPACE => Poids d'une arrête de c1 vers c2 : Nombre de lr + longueurs et scores alignements c1 + longueurs et scores alignements c2

— Étude des liens obtenus => Comment déterminer comment parcourir le graphe ? Comment choisir entre read / RC(read) ? => Tout produire, et pour chaque contig, regarder si contig ou RC(contig) produit le meilleur scaffold (le plus long / meilleur score / etc)

— Test de SSPACE sur Ecoli : Peu concluant (356 contigs => 154 scaffolds) => 2eme passe : 113 scaffolds

— Scaffolder semble peu intéressant => nécessité de trouver les k-mers des LR non présents dans les contigs

— Jellyfish : Permet de compter / exporter les k-mers dans un fasta

— Combiner Jellyfish + GkA => Visiblement 10h30 pour trouver tous les k-mers des LR non présents dans les contigs => Besoin de + rapide

2.57 Jour 72

— Mauvais résultats de SSPACE peuvent venir d'Abyss qui ne coupe par les reads / contigs au niveau des régions répétées

— Installation et test de CLCbio + SSPACE => Pas efficace

— Rerun SSPACE + Abyss avec les paramètres de Karlsson => Bien pour ADP1, bof pour Ecoli avec tous les jeux MinION

— Filtration des courts contigs Ecoli (< 1000) et rerun de SSPACE => Beaucoup mieux pour Ecoli

— Qualité / Couverture des scaffolds SSPACE très bonne => Vraiment nécessaire de remplacer les N par les bases des LR ?

— Installation et tests des GkA => Construction très longue + impossible de chercher un k-mer à partir de sa séquence

— Bug dans GkA : Quand on recherche un k-mer de taille > 63, il apparaît dans tous les reads, même ceux composés uniquement d'une même lettre répétée

2.58 Jour 73

— SeqBio

2.59 Jour 74

— SeqBio

2.60 Jour 77

— Test PgSA : Beaucoup trop long à construire

— Alignement des scaffolds produits par SSPACE avec BWA : Très mauvaise qualité => Probablement dû aux N ajoutés dans les gaps entre les contigs

— Semble y avoir peu de différence dans les résultats lors de l'utilisation de plusieurs jeux ou d'un seul jeu de MinION, plus de MinION permet juste de couvrir plus (pour SSPACE)

— Amélioration du code du Scaffolding

— Remplissage des gaps par les bases des LR sur un exemple (ADP1) => Aussi bon que SSPACE avec dnadiff, BIEN MEILLEUR avec BWA

— Test parcours du graphe sur Ecoli => Ne fonctionne pas

— Le scaffold produit par SSPACE ne commence pas par le contig mappé en position 1, mais scaffolds marqués comme "circulaires" quand c'est le cas

— Déterminer quel est le prochain contig : Avec le plus petit gap ? => Non

— SSPACE n'ordonne pas toujours bien les contigs, nécessité de mieux filtrer / mieux parcourir => Étudier l'ordre des contigs et les liens de BLASR

2.61 Jour 78

— Installation et tests sur les CGkA

— Correction parcours du graphe => Fonctionne avec Ecoli

— Travail sur le parcours du graphe avec exemple graphe Ecoli afin de déterminer un parcours meilleur que SSPACE

— Bug lors de la construction du graphe : Certaines arrêtes ne sont pas ajoutées => Problème vient du tableau de LR, qui peut se collisionner avec les tableaux d'autres contigs => À corriger

— Nécessité de trouver le contig le plus à gauche sur le Gen. Ref. non encore exploré pour faire le parcours ? => Semble impossible si scaffold circulaire

— alMin = 3000 semble permettre de bien lier chaque contig à son successeur direct => Recalcul des résultats SSPACE + nous => On semble être bien meilleur en utilisant BWA, pour ADP1

— Question : Pourquoi 284 scaffolds lorsqu'on ne filtre pas les < 1000 des 366 contigs Ecoli => Car SSPACE sort sous forme de scaffold les contigs non utilisés lors du scaffolding

— Problème : Même avec notre technique, on ne mappe pas en position 1 sur le génome de référence => Corrigé au jour 80 ; on ne choisissait pas la séquence RC du contig si le scaffold démarrait par un contig RC

2.62 Jour 79

— Question : Comment comparer deux génomes ? => dnadiff semble splitter le génome query en plusieurs parties => Si on split en moins de partie que SSPACE on est bon !

— Comparaison résultats dnadiff entre nous et SSPACE en fonction de cov / qual et nombre de parties alignées (moins il y en a mieux c'est) => On couvre mieux, un peu moins bonne qualité, et moins de parties alignées

— Comparaison résultats BWA entre nous et SSPACE en fonction de cov / qual => On a une bien meilleure qualité

— Recherche sur comment obtenir consensus de plusieurs séquence => Avec clustalw2 ? => Visiblement non car sort des caractères non-ACGT

— Correction bug génération scaffold avec Ecoli : Car le Ecoli-rccontigs.fa avait les contigs sur plusieurs lignes

— Bug du tableau des LR corrigé : Le problème venait de l'initialisation du tableau conseq servant à faire les consensus, avec i au lieu de size comme indice

— Débugage : Bug lors de la sortie des séquence des scaffolds, utilisation de valgrind => On copiait le
0 trop loin

— Travail sur les CGkA pour pouvoir chercher par séquence et non par position => Calcul nombre de k-mers des LR présents dans les contigs SR => Très peu

— alMin = 3000 lie bien chaque contig à son successeur direct, mais produit plusieurs scaffolds, nécessaire de les relier entre eux en raboutant, et d'obtenir un scaffold unique

2.63 Jour 80

— Calcul du nombre de k-mers des LR présents dans le génome de référence => Très peu

— Calcul du nombre de k-mers des LR présents dans les SR => Très peu

— Recalcul des résultats et retests : Si on part du même contig de début que SSPACE, on obtient un scaffold de meilleure qualité

— Tests SSPACE / GA sur ADP1 avec 1DR => Mauvais résultats

— Lecture LSCPLus (Correction LR avec alignement SR) => Intéressant en utilisant nos extensions pref / suff ?

— Recherche dans code SSPACE : Comment choisit-il le premier contig ? => Il prend le plus long, fait les scaffolds, et une fois les scaffolds produits, tente de les relier (là, il peut potentiellement prendre le RC d'un scaffold si cela peut permettre la création d'un lien)

— Modification du script de conversion de LSCPlus => Ne marchait pas bien

— Test de LSCPlus sur ADP1, M12D => Mauvais résultats

2.64 Jour 81

— Test de LSCPlus sur tout ADP1 => PAS ASSEZ D'ESPACE DISQUE

— Etude des MAW dans ref / LR / SR => Calculs longs

— Etude estimation des gaps / overlaps VS réalité sur ADP1 => Pas toujours juste, d'où l'impossibilité de mapper en un coup avec BWA

— Test avec consensus lors de gaps avec notre méthode => Diminue la qualité si on compte les occ des lettres => Besoin de trouver une stratégie de consensus plus efficace

— Étude des lens / scores nous vs SSPACE => Impossible de déterminer pourquoi ils sont différents, car les mêmes valeurs sont ajoutées aux liens => Besoin de refaire une recherche afin de pouvoir avoir les mêmes résultats que SSPACE en prenant alMin = 1500 TODO
— Correction de bugs dans le code de la GA (on ne remplissait pas le tableau de nom des contigs aux indices RC)
— Comparaison nous / SSPACE sur un autre génome (Yeast) => Résultats semblent similaires à SSPACE (à raboutage prêt)
— Tests sur SSPACE en autorisant les hits à ne pas être uniquement préfixe / suffixe => Aucun changement
— Yeast est en fait composé de 30 séquences de "référence" différentes => On devrait obtenir 30 scaffolds, mais SSPACE ne produit pas de bons résultats
— Tests de notre méthode sans le filtre SSPACE, uniquement en définissant alMin, idMin, gapMin => Résultats semblant être plus "continus" qu'avec le filtre sur Ecoli alMin = 1500, égaux au résultats avec filtre avec alMin = 3000 => MAIS le filtre sert à garder le meilleur alignement et non le premier de la liste en cas d'overlap trop important, et est donc utile

2.65 Jour 82

— Implémentation du raboutage des scaffolds
— Corrections sur le code (rev-comp d'une séquence, ...)
— Tests finaux : On est meilleurs quoiqu'il arrive (sur Ecoli / ADP1)
— Téléchargement d'un nouveau génome (Arabidopsis)
— Test de LSCPlus sur plusieurs jeux de données : Mauvais résultats

2.66 Jour 83

— Implémentation du "filtre" de SSPACE pour notre GA (différent de celui de SSPACE mais semble correct)
— Test assemblage des 64-mers LR non présents dans les contigs SR => Pas concluant
— Test avec notre méthode + répétitions => Pas concluant
— Réunion w/ TL => Besoin de chercher des k-mers espacés
— Test de comparaison des k-mers LR / SR / Ref / ... avec k = 32 => Toujours peu concluant

2.67 Jour 84

— Assemblage des SR Arabidopsis avec Abyss (très long...)

— Recherche sur l'extraction de spaced k-mers

— Test alignement LR -> contigs avec BWASW => Comparable à BLASR en termes de temps, trop difficile de parser le .sam

— Comparaison assemblage NaS vs Gen. Ref. avec dnadiff => NaS est toujours bien meilleur

— Recherche des 16-mers, 8-mers des LR dans les contigs LR => 16+-mers espacés (avec un 8-mer de chaque côté) semble la meilleure option

— Test de Miniasm => Moins bon que SSPACE / Nous

— Mapping avec minimap => Presque instantané

— Recherche sur les résultats de Minimap, pour voir si on peut scaffolder => Visiblement non

2.68 Jour 85

— Tests sur Minimap avec différents paramètres => Pas concluant

— Test assemblage des k-mers LR non présents dans les contigs SR => Pas concluant

— Séminaire

— abyss-pe long sur arabidopsis car needed subsampling => subsampled à 12M SR mais toujours très long + plante quand on lance d'autre processus

— Nouvelle idée : Assembler les k-mers des LR non présents dans les SR, et mapper les contigs obtenus assez longs sur les contigs SR => Pas concluant actuellement car trop de k-mers uniquement LR, mais peut être intéressant si on arrive à trouver un faible nombre de k-mers non présents ?

— Recalcul des k-mers LR présents dans les contigs des SR car erreur lors d'un test => Toujours mauvais résultats

— Recherche d'un nouveau génome sur lequel tester

— Test de DALIGNER => Instantané, mais comment voir les résultats ?

2.69 Jour 86

— Réunion w/ TL & AL => Pistes de travail pour l'indexation de spaced k-mers

— Assemblage des SR Arabidopsis avec Minia, car impossible avec Abyss => Ne produit aucun contig correct

— Problème ! ! ! => Les SR arabidopsis sont en fait trop courts !

— Génération de SR arabidopsis avec ART, puis test nous vs SSPACE avec assemblage Minia (car SSPACE trop long) SSPACE => Mauvais résultats avec al = 3 000 / overlap = 5 000 => car LR = PacBio SSPACE => Retest avec al = 1 500 / overlap = 9 000 => Mieux, mais toujours pas d'unique scaffold Nous => On fait pire que SSPACE => BESOIN DE REVOIR COMMENT CHOISIR ENTRE F ET RC POUR DEBUTER UN SCAFF => Mauvais résultats à cause de la taille des contigs Minia ?

— Installation et tests de Graphmap => Ne peut trouver qu'un mapping par LR => Inintéressant

— Recalcul recherche des k-mers LR avec RC => 2 x plus fois 64, 1,75 x plus avec 32, 2x plus avec 16, presque tout avec 8

— Étude des résultats de DALIGNER => Pas concluant, ne permet pas de relier convenablement

— Recherche des k-mers NaS dans contigs SR => Quasiment tous présents

— Nouvelle idée : Prendre un LR, rechercher tous ses k-mers (non chevauchant) de taille 128 dans l'ensemble des MAW de taille 128 des SR => Aucun résultat => Au final, revient à chercher les k-mers des LRs dans les k-mers des SR => Donc reprend notre idée de spaced k-mers

— Assemblage avec Abyss (il ne faut aucun autre process en même temps)

— SSPACE vs Nous : On fait n'importe quoi

2.70 Jour 89

— Étude de notre scaffolding, car résultats très différents de SSPACE sur Arabidopsis

— SSPACE produit également de mauvais chemins, dû au mauvais jeu de données SR, car trop faible couverture ? => En effet, Karlsson recommande un coverage x1000, et le papier SSPACE également, alors qu'on était à 49

— Relecture papier GkA

— Tests sur MWE pour les spaced-GkA => Semble marcher

2.71 Jour 90

— Regénération de SR avec ART (100x coverage) + assemblage Abyss

— Test SSPACE et Nous sur le nouveau jeu de données

— => Impossible, assemblage Abyss interminable

2.72 Jour 91

— Re-MWE sur spaced-GkA

— Étude sur comment faire un spaced SA => Simplement par tri des suffixes

— Test de LAST : Rapide, mais produit de mauvais liens + ne prend pas en compte les strands query / db

— Etude du code de GkA => Impossible de trouver comment remplacer le SA par un spaced-SA

— Reprise du code des tests de CGkA afin de rechercher des k-mers espacés

— Test de la recherche des k-mers espacés sur 8 -> 64, gap = 1

2.73 Jour 92

— Test de la recherche de k-mers espacés, gap = 2, 3

— Test de la recherche de k-mers espacés, gap variable (jusque 10)

— Test de correction de LR à l'aide de seeds + unitigs overlapant => Pas concluant avec Minia (unitigs trop courts) ni avec Abyss (trop peu d'unitigs)

— Étude du code de CGkA pour remplacer SA par spaced-SA => Impossible de trouver également

— Bug sur la recherche de k-mers espacés ? Pas les mêmes résultats avec k-mers contigus et 0-spaced-k/2-mers ...

— Bug corrigé => Recalcul de recherche des k-mers espacés

— Nouvelle idée : Mapper les SR sur les LR, s'en servir comme seed, et relier les seeds à l'aide des GkA

2.74 Jour 93

— Encore des bugs dans la recherche de k-mers espacés... => Correction

— Recalcul recherche k-mers espacés LR dans CSR => Pas concluant

— Construction CGkA des LR

— Test recherche des k-mers CSR dans LR => Pas concluant

— Test de reliage de seeds à l'aide des GkA => Bug, à reprendre

— Nouvelle idée : Recherche de spaced-k-mers de LR DANS les LR, afin de déterminer bons / mauvais k-mers

— Recherche des k-mers espacés LR dans CSR => Pas concluant

2.75 Jour 96

— PBJelly2 : Même idée, mapper LR sur CSR avec BLASR, mais impossible à installer / tester

— Travail sur le reliage de seeds à l'aide des CGkA / PgSA => Impossible avec des reads de longueur différente => Problématique

— Subsampling de l'ensemble de SR ADP1 en ne gardant que ceux de longueur 301

— Retest de reliage de seeds avec CGkA => Très rarement des overlaps parfaits à k fixé => Nécessaire de diminuer peu à peu la taille du k-mer recherché afin de trouver des overlaps => Donc nécessaire d'utiliser PgSA

— Construction de l'index PgSA, et étude des fonctions traitant les requêtes

2.76 Jour 97

— Réunion w/ TL & AL : Validation du spaced-GkA, idées pour la recherches de nouveaux spaced-k-mers, trimming des LR, ...

— Travail sur PgSA pour permettre de relier les seeds (tout l'aprem)

— Calcul des ensembles de k-mers des LR trimmés de 8

— Recherche des 64-0-10-spaced-k-mers, puis des 32 des non trouvés, etc => Bug car les k-mers non trouvés sont écrits dans le fichier en RC => Correction et rerun

2.77 Jour 98

— Recherche des 0-10-spaced-k-mers dans les 8-trimmed-LR => Pas concluant, donc erreurs pas en début-fin mais aléatoires

— Bug dans la production des 32-mers => Recalculer les k-mers présents dans les CSR (peu de changement donc osef)

— Poursuite du travail sur PgSA => Mauvais résultats car certains SR sont de mauvaise qualité => Correction des SR (avec Karect) et retest => Karect trop long, test avec Lighter => N'améliore pas beaucoup la qualité, seulement +0,2 % id

— Pour le reliage avec PgSA => Si on utilise des spaced-SA, on pourra potentiellement mieux relier (car tolérera qq erreurs de subs, parfait pour Illumina) => NON EN FAIT, car un spaced-SA sert à rechercher des k-mers avec des gaps entre les bases et non des k-mers avec des mismatches

— Recherche sur les spaced-SA => Un algo (DisLex) mais pas dispo, ou radix sort

2.78 Jour 99

— Test de reliage avec PgSA en utilisant les SR corrigés par Lighter => Ne change rien => Parfois dans jeu Illumina, présence de reads de faible qualité, qui font merder le reliage

— Test de perM pour comparer les k-mers : Trouve un peu plus de k-mers que précédemment, mais pas top non plus, prend en compte des erreurs de mismatches, et non des erreurs d'indels => Bons résultats avec des 16-mers, car tolère une erreur par 8-mer, et car presque tous les 8-mers sont trouvés sans erreur

— Test reliage de deux reads très éloignés avec PgSA => Pas très efficace, car toujours certains SR de mauvaise qualité

— Nouvelle idée : Pour le reliage avec PgSA, le faire avec l'ensemble de k-mers des SR ? Permettra par ex. de trouver un lien entre un SR et le milieu d'un autre SR => Fonctionne en effet bcp mieux niveau qualité (même descendant à overlap = 10) => Mais produit des liens plus courts, quoique agrandissable en baissant overlapMin

— Prog d'un algo donnant le SA pour un pattern nb match - nb gap donné

— Différenciation spaced-seeds de la littérature, et spaced-k-mers définis par nous

2.79 Jour 100

— Réunion w/ TL & AL => Rechercher des k-mers avec 20-21 bases solides, obtenir DisLex pour rechercher nos gapped-k-mers, continuer PgSA

— Recherche de k-mers avec 20-21 bases solides avec perM => Assez mauvais résultats, plus proche des 32-mers que des 16-mers

— Mail pour code source DisLex

— Test recherche de mot dans SSA et dans SA => Les SSA tels qu'on les a défini permettent bien de rechercher des motifs avec gaps, et non avec mismatches => GOOD

— Travail sur reliage avec PgSA (utilisation des RC des k-mers) => Segmentation fault : A cause de k-mers en double ?

2.80 Vavances de noël : Jours 101 - 117

— Retest génération PgSA avec RC 64-mers, en ne gardant qu'un exemplaire de chaque 64-mer => Bug toujours

— Génération PgSA avec RC 64-mers, en ne gardant que les 64-mers apparaissant au moins 2 fois => Works ! => Les "contigs" ainsi obtenus sont de bonne qualité, mais toujours difficile de relier 2 64-mers sans backtrack

— Recherche des 20-mers avec CgKA

— Recherche des 21-mers impossible : Car on ne peut charger en mémoire qu'un seul index, et diviser les k-mers à rechercher par deux => Donc recherche des 22-mers à la place

— Programmation du backtracking pour relier les seeds => Ne marche pas bien sur l'exemple visant à relier 2 seeds mappés sur un LR => Marche très bien, insta, 99% id pour relier 2 seeds provenant du gen. ref.

2.81 Jour 118

— Test de reliage de 2 seeds provenant du gen. ref. avec les SR 250bp de NaS => Identité de 100%

— Relier des seeds provenant du mapping SR sur LR marche en fait

— Retravail sur la fonction de reliage, afin de ne pas return 10000 fois

— Travail sur pblat, afin de produire un fichier contenant les seeds à relier => Fait, en reprenant le code de l'algo du stage

— Problème : Encore des seeds de mauvaise qualité, du à la fusion => Nécessité de consensus ? De prendre seulement le meilleur ?

— Solution retenue : Quand il y a des chevauchements, on ne garde que le seed avec le plus de matches => Permet de n'avoir quasiment que des seeds s'alignant à 100%, le plus faible = 99 %

2.82 Jour 119

— Problème : Certains k-mers des seeds ne sont pas présents dans le PgSA (sûrement parce qu'on a demandé au moins 2 occ de chaque k-mer avant la construction) => Tests avec le sous-ensemble de reads de longueur 301

— Même en retenant les seeds se mappant avec une bonne identité, toujours impossible de relier certains k-mers, semble-t-il à cause d'erreurs de mismatches (généralement on tombe sur le k-mer cible à UN SEUL mismatch près)

— Tentative de solution : Considérer les seeds comme liés quand on arrive sur un k-mer chevauchant le seed destination sur 2/3 de sa longueur => Semble bête par rapport à la solution du dessous

— Tentative de solution : Considérer les seeds comme liés quand on arrive sur un k-mer avec < n différences avec la cible => Beaucoup plus correct

— Idée : Utilisation d'un double PgSA (64 et 32 mers par ex) s'il est impossible de relier 2 seeds avec un k PgSA de taille k, on en prend un plus petit ? => Non, car les k des PgSA est variable

— Problème : Backtracking bug encore mais c seulement la prog

2.83 Jour 120

— Travail sur l'algo de reliage de seeds avec PgSA, correction de bugs, mise en évidence de problèmes

— Le backtracking fonctionne et retourne correctement

— La correction semble marcher au moins avec le jeu NaS

— Mise en évidence d'un problème si les seeds sont plus distants sur le gen. ref. que sur le LR => Perte de qualité et augmentation temps d'exécution

— Toujours des problèmes avec les free

— Besoin de plus de seeds, car certaines zones du LR sont peu couvertes, et donc perte de longueur sur le LR corrigé

2.84 Jour 121

— Travail sur algo reliage, notamment sur les free

— Idée : Utiliser des k-mers comme seeds => Impossible, car pas assez de longueur pour être précis, on peut donc avoir des placements ne concordant pas sur le LR et sur le gen ref (par ex, S2 à droite de S1 dans le LR, mais à gauche dans le gen. ref.), amenant à des impossibilité de reliure

— On peut en fait avoir le même problème avec des SR complets comme seeds, mais c sûrement plus rare

— Recherche de paramètres pblat

— Erreurs de malloc / free semblaient venir de algo.c, qui produisait mal les séquences à relier

— Programmation d'une fusion des seeds s'ils se chevauchent correctement au lieu de juste garder celui avec la meilleure qualité

— Utiliser des SR complets comme seeds va être difficile, car on impose une forte contrainte sur leur nombre de match, et peu de LR pourront donc être corrigés Idée => Utiliser des LR complets, récupérer le fichier "tolink", et le corriger ?

2.85 Jour 123

— Lors du mapping SR -> LR pour obtenir les seeds, mapper les SR ou les SR + leur RC n'amène pas au même résultat, bizarre

— Les free provoquent toujours des erreurs si on copie avec mlen dans algo.c, mais avec rlen ça passe

2.86 Jour 124

— Préparation prochaine réunion

— Mapping des SR sur tous les LR 1D / 2D avec minScore 150 pour compter les seeds

— Téléchargement et test de Quorum (correction rapide de SR) => Peut être prometteur, car plus besoin de minScore, et seeds obtenus s'alignent à 100 %, et très rapide

— Retravail sur l'algo reliage, car bugs avec malloc / free, encore => Tous retirés, semble ne plus poser de problèmes

— Nouveau problème : Quand les seeds sont plus disants sur les LR que sur le gen. ref., il est alors possible que les SR soient distants sur le LR alors qu'ils s'overlappent sur le gen. ref.

— => Solution + amélioration : Calcul des overlaps entre les seeds avant de stocker dans le fichier

— Une fois les overlaps calculés, le premier exemple fonctionne bien plus rapidement si on laisse le 25eme seed, mais celui-ci est toujours erroné

— En fait non, si on ne met aucune condition de score, le 1er exemple bug juste au milieu, mais si on le sépare en deux, les deux parties s'alignent bien à 100 %. De même, si on retire les deux seeds qui posent problème (pck autour de 900k sur le gen. ref., alors que les autres autour de 300k), on obtient une seule partie s'alignant à 100 %

2.87 Jour 125

— Mapping des SR corrigés sur tous les LR 1D / 2D pour compter les seeds => Moins de LR avec seeds après correction, mais plus de seeds par LR, donc plus intéressants

— Séparation des tests SR bruts / corrigés dans le PDF réunion

— Recalcul des 64-mers des SR NaS corrigés, avec ajout des RC des SR NaS corrigés => Ne change rien aux résultats précédents

— Recherche sur comment faire la partie non reliable du 1er ex => Skipper le seed non reliable (44 -> 45) du 1er exemple et continuer ne change rien, toujours impossible de relier 44 aux suivants => Mapper les SR + leurs RC sur le LR ne change rien, 44 et 45 ne se relient pas => Recherche de tous les 64-mers de la partie du gen. ref. qui cause l'impossibilité de reliure => Bizarre car tous présents dans notre ensemble de 64-mers servant à générer le PgSA => L'EXEMPLE 1 DEVRAIT PRODUIRE UN LR UNIQUE, plante sûrement à cause de la limitation du nombre d'appels récursifs

— Tests avec l'ensemble complet de SR, après correction => Ne marche pas sur le 1er exemple, car présence d'erreurs dans les seeds

— Automatisation de la séparation du LR synthétique en plusieurs parties quand certains seeds ne peuvent pas être reliés

— Recherche paramètres pblat pour faire disparaître les seeds mappés dans les 900k sur l'exemple 2 => Augmenter stepSize semble les faire disparaître, mais les seeds à relier sont alors plus éloignés, ce qui cause une segfault car trop de backtracks

— Implémentation d'un nombre max de retours arrières + recherche d'un seuil => Problème : Semble impossible de relier des seeds trop distants, à cause d'une seg fault quand trop d'appels récursifs

— => Besoin de revoir l'implémentation de la fonction de backtracking

— Quand tous les seeds se mappent sur le LR dans l'ordre inverse d'apparition sur le gen. ref. => Pas de soucis en fait !

2.88 Jour 126

— Dépression.

— Test de notre méthode (avec les SR corrigés) sur le jeu test NaS => Marche parfaitement

— Test de NaS (après galère à faire marcher) sur le jeu test NaS => Résultats un peu meilleurs, mais 6 fois + long

— Tentative de modification de l'algo pour relier => Gros bugs de partout, dur de revenir à une version qui marche

— Amélioration de l'algo pour relier, dans le cas où il n'y a pas d'ambiguité (un seed ne peut être étendue que par un seul k-mer) : plus d'appels récursifs => Permet d'améliorer le temps d'exécution => Besoin de retravailler un peu, dans le cas où deux seeds sont difficiles à relier (car ici, on baisse le chevauchement nécessaire et on fait le backtrack à partir de l'extension maximale du seed, et non du seed lui même)

2.89 Jour 127

— Ajout d'un filtre de taille max de l'extension => Maintenant, il est possible de relier tout le LR du 1er test en une seule partie ! => Nécessité de fixer cette taille max à la taille du template ? => Effectivement, semble bien marcher

— Reprise des tests avec le filtre taille extension et comparaison nous vs NaS sur exemple LR1/2/3 => On le défonce

— Script pour tester sur un grand ensemble de LR

— Test sur M12D : 44min, nb reads / fragments = 566 (contre 602 LR initiaux) avlen = 4 906 (contre 5 197 LR initiaux), avid = 99,84 % => Notre solution marche vraiment bien !

2.90 Jour 128

— Mapper les SR sur chaque LR séparément et sur tous les LR d'un coup ne produit pas les mêmes résultats => Une méthode fast (tout d'un coup) et une méthode sensitive (chaque LR séparément) ?

— Programmation de fonction extension à gauche et à droite => Mange à peine plus de temps (10 sec) et permet de gagner une avLen de quasiment 1k, et d'être plus long que NaS, tout en gardant une excellente identité et autant de reads fragmentés, mais plus petit cumSize que NaS (quand on mappe tout d'un coup)

— Retravail sur sortie de la méthode : Les LR corrigés sont maintenant sortis sur une seule ligne

— Réunion w/ TL & AL => Validation de la méthode, besoin de commencer à écrire article, de trouver compromis entre tout mapper et mapper séparément, étudier k-mers des SR corrigés et bruts

2.91 Jour 130

— Préparation diapos pour réunion MASTODON

— Début répétition réunion MASTODON

— Test de notre méthode sur un autre jeu de données => Sur reads 1D, légèrement moins bonne qualité que NaS (-0,25 %), mais bien meilleures longueur moyenne et taille cumulée

— Reprise des tests sur M12D car résultats pas identiques à ceux notés sur le pdf => On corrige moins de reads si on étend à gauche et à droite : wtf ? => En fait non, après retest ça marche, mais bizarre

— Quand on mappe tous les SR sur tous les LR d'un coup => Si il n'y a qu'un seul seed sur un LR, alors on l'étend et on sort cette extension comme correction => Permet de corriger (à 1 ou 2 près) autant de LR que NaS

— Test de correction (sur M12D et M41D) en passant le nombre d'erreurs tolérées (dans algo.c et fichier reliage) à 0 => Sur M12D, 2 reads fragmentés en plus, - 50 avLen, - 6 217 cumSize, + 0,03 % avId => Sur M41D, 8 reads fragmentés en plus, - 146 avLen, - 23 422 cumSize, + 0,04 % avId => On garde

2.92 Jour 131

— Test sur tous les LR ADP1 => Rapide et très efficace sur les reads 1D, long sur les reads 2D car très nombreux => test arrêté, mais bonne qualité des résultats produits jusque là

— Test sur les autres jeux de données NaS (Ecoli et Yeast) => Pareil, beaucoup trop de LR => Nécessité de prendre un échantillon de ces jeux de données pour tester

— Amélioration diapos MASTODON (matinée)

— Réunion Stringmasters (Aprem)

2.93 Jour 131

— Test sur Ecoli et Yeast avec des échantillons de 500 LR => Bons résultats

— Retirer la liste des k-mers déjà visités => produit des reads plus fragmentés et est + coûteux en temps

— Début écriture article

— Test tuning paramètres PBLAT pour ne plus fragmenter les reads => Ne donne rien => Comment ne plus fragmenter les reads ?

2.94 Jour 132

— Poursuite écriture article (Intro, PgSA overview, début notre méthode, workflow notre méthode)

— Étendre les deux seeds (à gauche et à droite) et chercher un overlap quand il est impossible de les relier => Semble ne pas marcher, très peu d'overlaps

— Correction de l'extension si seed unique : On étendait seulement à gauche

2.95 Jour 133

— Recalcul des résultats des tests maintenant qu'on étend aussi à droite si seed unique => 1k + long en moyene

— Création dépot Git

— Comparaison des LR initiaux / LR corrigés => Rien de commun entre reads NaS et templates d'origine (4 k-mers) => Rien de commun entre nos reaads et templates d'origine (33 k-mers)

— Poursuite écriture article (Fin explication notre méthode)

— Test en réduisant la taille de l'overlap nécessaire pour fusionner 2 seeds (63 => 32) => Seulement très peu de LR fragmentés en moins, et baisse de qualité => Quand overlap trop court et qu'on ne garde qu'un des deux seeds, garder le plus long (donc précédemment étendu) semble permettre une meilleure qualité finale

— En fait, certains seeds qu'on tente de relier se chevauchent, d'où l'impossibilité

— Calcul des résultats NaS pour Ecoli et Yeast

2.96 Jour 136

— Recherche sur comment ne plus fragmenter la correction (tuning paramètres BLAT, etc) => Pas fructueux

— Test alignement SR -> LR avec BLASR : 5min en 8 threads avec ADP1, M12D, 50 LR avec seeds de plus qu'avec BLAT

— Test correction avec alignement BLASR (score le plus bas = meilleur) => Semble produire encore plus de LR fragmentés que BLAT

— Réunion w/ TL

— Comptage distribution des k-mers LR bruts / corrigés => La correction tend à faire apparaître plusieurs fois les mêmes k-mers

— Installation et tests de Commet (comparer des ensembles de reads par relation de similarité, 2 kmers partagés = reads similaires) sur NaS et notre méthode => Permet de déterminer de quel ensemble de LR bruts vient un ensemble de LR corrigés

— Comparaison des k-mers communs LR bruts / corrigés, avec CGkA, pour vérifier les résultats de Commet => Très peu en commun

2.97 Jour 137

— Retest tuning paramètres BLAT => Rien à faire pour fragmenter moins, que ce soit en augmentant ou droppant les seuils d'accuracy / qualité

— Comparer les k-mers des LR bruts / corrigés revient à comparer les k-mers des LR bruts / contigs SR, car LR corrigés formés de SR

— Comparaison k-mers fréquents (> 1 fois) des LR bruts / corrigés => Pas concluant, seul la moitié des k-mers fréquents des LR bruts sont fréquents dans les LR corrigés (jusqu'à 16-mers), et très peu à l'inverse

— Tentative réinstallation BLASR, pour pouvoir sortir en BAM => Impossible, complètement impossible quoiqu'on fasse

— Vérification, après correction, on couvre autant (même un peu +) le gen. ref. que NaS

— Vérification en comparant nos reads aux reads NaS par comparaison de k-mers avec CGkA ou directement avec Commet => Quasi 100 % de similarité

— Test : Quand impossible de relier deux seeds, on essaye de relier le seed de gauche au suivant jusqu'à ce que ce soit possible => Plus aucun LR fragmenté, gain de avLen mais baisse de qualité (- 0,2 %)

— Test : Ne considérer que les bases qui matchent (et donc ne plus appliquer de malus pour les mismatches / indels) lorsqu'on détermine quel est le meilleur de deux seeds s'overlappant => Pas concluant, produit des résultats plus fragmentés

— Commet plante quand k = 64, mais indique que tous les LR NaS sont similaires aux notres et vice-versa

2.98 Jour 138

— Recherche des k-mers des LR bruts dans les LR corrigés => Pas la même chose que chercher les k-mers des LR corrigés dans les LR bruts ? Bizarre

— Remodification paramètres PBLAT => Pas d'amélioration, quoique soit le paramètre modifié

— Retest d'installation de BLASR => TOUJOURS IMPOSSIBLE

— Test alignement SR -> LR avec Bowtie2 => Pas satisfaisant

— Test mapping avec BLASR et sortie SAM obsolète (M12D) => Possible de corriger plus de LR (30aine), de moins fragmenter les résultats, d'obtenir une qualité très similaire (0,01 supérieure), mais un peu plus long à run + plus long à mapper => Tune les paramètres pour corriger encore plus ?

— Plus possible d'obtenir la longueur du template direct dans le fichier de mapping si on mappe avec BLASR, donc besoin de borner la taille max d'une extension autrement, ou de récupérer taille du template dans algo.c, en recherchant la seq à partir de son id, mais c'est long

— Limiter la taille de l'extension à 130 / 100 * gapSize => Pas concluant, beaucoup trop de LR fragmentés, retour à la limite = longueur tpl

— Test de mapping avec BLASR (M41D) => Possible de corriger plus de LR (430), de moins fragmenter, d'obtenir une qualité très similaire, mais plus long à run + plus long à mapper

— Bugs de mémoire quad utilisation de BLASR en mappeur ? ? ?

2.99 Jour 139

— Début présentation Génoscope

— Test sur différentes limites de taille max extension

— Correction de bugs dans l'algo (l'extension gauche provoquait des fragmentations inutiles si on bornait la taille de l'extension avec un dérivé de la distance entre les 2 seeds, maintenant elle est fait à la fin, et l'extension sans ambiguité pouvait provoquer la production d'un mauvais LR) => Gain de qualité

— Test tuning des paramètres de BLASR => abaisser minMatch permet de corriger un peu plus de LR, mais fragmente un peu plus

— Test tuning des paramètres de BLASR (maxScore = 0 + minMatch = 10) => Mauvais résultats, score trop haut

— Test tuning des paramètres de BLASR => Augmenter maxScore permet de corriger un peu plus de LR, mais fragmente beaucoup plus

— Test BLASR, en gardant les alignements complémentaires => Permet de corriger plus de LR, tout en fragmentant moins, quels que soient les autres paramètres

— Test BLASR, en gardant les 20 meilleurs alignements (au lieu de 10) => Pas concluant, corrige juste un peu plus

— Les bugs de malloc / free / etc dans l'algo venaient de BLASR qui sort des alignements de taille < 64 => Corrigé avec ajout d'un paramètre, mais encore quelques bugs, car encore des alignements de < 64 sont sortis

— Test de NaS sensitive avec M12D : Au max 490 LR corrigés => On doit faire au moins aussi bien

— Test de NaS sensitive avec M41D => On fait mieux avec les paramètres par défaut

— Test tuning des paramètres avec M41D (maxScore -150 et minMatch 10) => Trop de fragments

2.100 Jour 140

— Meilleurs paramètres : Garder les paramètres par défaut et conserver les alignements complémentaires, semble permettre de corriger plus de LR que NaS

— Implémentation de la récupération de la longueur du template dans son identifiant => Deux fois plus rapide !

— Amélioration des diapos (ajouts de onslide, de schémas, etc)

— Test en ne diminuant pas le nombre de backtracks en sortie de fonction, pour ne pas perdre 10min sur certains couples de seeds => Permet d'obtenir sensiblement les mêmes résultats, plus rapidement

— Recalcul de tous les temps d'exécutions BLASR avec la longueur du template récupérée dans son id, et la nouvelle limitation de backtrack => Même résultats, mais bien plus rapide

2.101 Jour 143

— Test : Relier le seed au suivant et keep going, en utilisant BLASR => Légère baisse de qualité, production de moins de LRs, pareil qu'avec BLAT

— Idée : Assemblage des k-mers fréquents (> 1) des LR avec PgSA => Peuvent à peine être étendus, ambiguïté direct Mais les k-mers récupérés ont une identité moyenne de 99,46 et couvrent au max 80 % du génome => Pas concluant

— Idée : Aligner les LR entre eux pour en récupérer les parties correctes => Pas concluant, difficile de trouver des alignements autres que d'un LR sur lui même

— LoRDEC semble être plus rapide que nous, test : Effectivement rapide, mais incapable de corriger les LR MinION

— Idée : Récupérer les "k-mers" de taille 250 des LR, les corriger, et les utiliser comme les SR de notre méthode hybride => Ne semble pas concluant, Quorum ne peut pas les corriger

2.102 Jour 144

— Récupération de tous les 64-mers des LR et analyse de leur qualité => 80 %, on ne peut rien en faire

— Retest de récupération de "k-mers" de taille 250 => En fait, produit beaucoup trop de ces k-mers, fichier de 87Go

— Avec des k-mers de taille 250 apparaissant plus d'une fois : Seulement 12 LR présents, pas intéressant

— Avec des k-mers de taille 100, apparaissant plus d'une fois => Pas intéressant non plus, pas assez de k-mers, puisque déjà pas assez avec k = 64

— Reprise de l'idée d'alignement des LR entre eux pour récupérer les parties correctes, avec un script permettant de trouver les alignements non-self => Très long, peu d'alignements, les séquences déduites ne s'alignent pas toutes, et avId = 90 => Pas concluant

— Tuning des paramètres de BLASR pour minimiser les LR fragmentés => minMatch à 10 semble permettre de corriger + et fragmenter -, si on garde les alignements complémentaires

— Reprise du code algo.c pour déterminer quel seed prendre en cas d'overlap => prendre celui avec le meilleur score produit des LR plus fragmentés et de moins bonne qualité, donc on reste comme on est, on garde le plus long

— Test en utilisant directement les k-mers comme seeds => On corrige moins de LR, pas concluant

— Test sur Yeast avec BLASR => Légère perte de qualité par rapport à BLAT, mais + de LR corrigés

2.103 Jour 145

— Récupération des ensembles de LR fragmentés par la correction, et test de nouveaux paramètres pour les défragmenter

— Reprise de algo.c pour le scoring quand on fusionne 2 seeds, afin de mettre le score réel Test avec choix du seed avec le meilleur score en cas d'overlap trop court => Produit d'aussi bons / meilleurs résultats qu'avant et est plus logique => on garde

— Recherche d'outils de selfcorrection de LR, et retouche de l'article

— Test de skip en cas de seeds impossibles à relier (on essaye de lier celui de gauche au prochain à droite and so on) => Bons résultats sur M12D (2 LR en moins seulement), mais peut être extrêmement long si le "mauvais" seed est le premier du LR, par ex sur M42D

— Test de correction des 64-mers avec quorum => Inutile, aucun gain

— Test d'alignement des k-mers en augmentant le nombre de hits à garder => + de 1h20 en gardant 100 hits, et seulement 6 seeds de plus => Pas concluant

— Test d'alignement des seeds classiques en augmentant le nombre de hits à garder => Permet de corriger un peu plus de reads, mais peut amener une légère baisse de qualité (pour M41D)

— Test d'alignement en augmentant bestn + en diminuant minMatch => Semble corriger encore un peu plus et fragmenter encore un peu moins, mais 2 fois plus long à run => Pas intéressant

2.104 Jour 146

— Réunion nouveaux arrivants

— Traduction diapos en anglais

— Test en diminuant maxScore => Corrige moins de LR et fragmente à peine moins

— Test en diminuant la longueur nécessaire d'un overlap pour fusionner deux seeds => Ne change rien pour M12D

— Idée : Quand on a des LR fragmentés, tenter de relier les fragments avec une nouvelle passe de l'algo => Semble marcher pour quelques LR, mais pas pour la majorité => Intéressant ?

— Problème : Le nombre max de backtracks est parfois un peu dépassé => En fait non, on return juste direct un résultat négatif dans ce cas

— Test : Nos LR et les LR NaS se mappent aux mêmes endroit, à quelques bases près => On corrige bien

2.105 Jour 147

— Poursuite écriture article (résultats, présentation datasets, reprise schéma, ...)

— Test en passant le seuil min d'overlap à 32 => Ne semble pas changer grand chose, car la fragmentation sera principalement causée par le dépassement du seuil de backtracks ou des seeds mal mappés, mais on garde car plus logique que le seuil arbitraire de 40

— Idée : Scaffolder les assemblages fragmentés avec des reads courts => SSPACE n'y parvient pas

— Cleanup des résultats des tests sur le fichier réunion => Conversation des résultats parlant seulement

- Comparaison des runtimes pour l'alignement seulement BLAT / BLASR => BLAT est quasi instantanné, d'où l'augmentation de runtime avec BLASR
- Run de correction sur machine perso : L'exec sur serveur est effectivement plus longue à cause des processus qui runnent (57min vs 1h30 pour M41D)
- Besoin de tous les LR pour réaliser l'assemblage sur ADP1 => Correction de tous les ensembles
- Optimisation : En fait, si on veut garder le seed avec le meilleur score en cas d'overlap trop court, overlapper les seeds s'ils se chevauchent sur une longueur >= 31 semble produire de meilleurs résultats que si on fixe cette longueur à 63
  => Impossible de finir la comparaison car les tests plantent sur le serveur crihan, mais vrai sur les ensembles testés
- Test assemblage nos LR vs NaS => On semble être assez mauvais, 5 contigs et seulement 1/2 du génome couvert
- NaS corrige un peu plus de reads en mode sensitive => Car il utilise tous les SR en seeds, alors qu'on utilise les SR corrigés, et qu'il y en a donc un peu moins
- Nous vs NaS : On perd seulement 0,3 % d'identité
- Comparaison NaS sur le site vs ce qu'il produit vraiment => Cohérent

2.106 Jour 150

- Test de notre méthode avec d'autres valeurs de k
  128 : 487 LR dont 44 fragmentés, avLen = 4 852, cumSize = 2 629 486, avId = 99,22 %, temps = 25 min 42
  64 : 492 LR dont 18 fragmentés, avLen = 5 449, cumSize = 2 822 627, avId = 99,96 %, temps = 21 min
  32 : 492 LR dont 23 fragmentés, avLen = 5 293, cumSize = 2 816 039, avId = 99,18 %, temps = 13 min 18
  70 et 50 : baisse aussi
- Test assemblage de NaS_{FAST} => *Unseulcontigaussi*

- Stringmasters (jour 1)
- Test assemblage nos LR avec les bons paramètres => Il semble être possible d'arriver à obtenir un contig unique qui couvre tout le génome

2.107 Jour 151

— Stringmasters (jour 2)

2.108 Jour 152

— Stringmasters (jour 3)

— Lors de la fusion de 2 seeds, le score calculé n'était pas le bon => Modification

— Tests pour obtenir mêmes résultats sur serveur / machine perso => Impossible même avec les mêmes fichiers / indexes => À cause des fuites de mémoire ?

— Test sur tous les LR d'un coup => Environ 10k LR corrigés en moins, qualité similaire, longueur supérieure, 6x plus rapide que NaS

— Comparaison correction tout d'un coup VS 1D puis 2D sur run 1 => Corriger séparément permet de corriger plus de LR, et de moins fragmenter mais les résultats en termes de qualité son similaires

2.109 Jour 153

— Comparaison sur tous les LR avec overlap de seeds = 63 vs overlap de seeds = 31 => 63 est très légèrement meilleur, mais + de LR fragmentés

— Script retrait des LR fragmentés du fichier final => Bug à cause des segfault dans l'algo

— Assemblage des LR corrigés run par run en ne gardant que les LR non fragmentés => Possible d'avoir un seul contig en tunant les paramètres

— Assemblage des LR corrigés run par run avec les LR fragmentés => Impossible d'avoir un seul contig

— Assemblage des LR corrigés d'un coup en ne gardant que les LR non fragmentés => Possible d'avoir un seul contig en tunant les paramètres

— Assemblage des LR corrigés d'un coup avec les LR fragmentés => Possible d'avoir un seul contig en tunant les paramètres

— Comparaison runtime assemblage : Nous = environ 1h, mais pour NaS aussi

2.110 Jour 154

— Recherche de la segfault dans l'algo => À cause d'alignements de mauvaise longueur sortis alors qu'ils ne devraient pas l'être

— Création d'un script python pour filtrer les alignments trop courts

— Alignement des raw LR avec Last => Conversion en SAM incorrecte, impossible de récupérer les alignements correctement

— Test d'augmetation du paramètre besthit quand on corrige tous les LR d'un coup => Permet effectivement de corriger plus de LR

— Modifications article

— Comparaison résultats machine perso / serveur avec la correction des segfaults => Réglé

— Correction d'un autre bug de segfault dans l'algo => Car les seeds peuvent être plus longs que 1024 !

— Récupération des résultats en filtrant les alignements trop courts + correction segfault + en définissant l'overlap min à 63

2.111 Jour 157

— Test assemblage avec les nouveaux jeux de LR corrigés => Résultats pire qu'avant à cause des LR fragmentés

— Assemblage des LR corrigés par NaS

2.112 Jour 158

— Poursuite assemblage LR corrigés / LR NaS

— Remplissage des tableaux de l'article

— Assemblage des LR corrigés par notre méthode => Très mauvais résultats

— Recherche de nouveaux paramètres Canu pour assembler nos LR => Un peu mieux avec les seuils d'erreur à 0.15, mauvais avec les k-mers à 22 => Trouvé, marche même avec les LR fragmentés

2.113 Jour 159

— Tests d'assemblage, ajustement des paramètres => Max semble être à 99,95 % de couv avec ADP1

— Test correction + assemblage full Coli => Les LR synthétiques produits semblent ne pas couvrir suffisamment le génome de référence

2.114 Jour 160

— Tests correction Coli en augmentation bestn => On couvre le génome à 100%

— Avance écriture article (paragraphe mapping / assemblage / conclusion)

— Ajout tableaux résultats sur diapos Génoscope

2.115 Jour 161

— Réunion w/ TL & AL => Présentation des nouvelles idées, validation, rapide

— Test assemblage Coli => Impossible avec les paramètres actuels, alors qu'on couvre bien à 100 %

— Reprise article pour le passer au format Jobim

— Test tuning des paramètres pour bien assembler ADP1 (valeur de k, nb de backtracks, taille min de chevauchement, ...)

— Il est nécessaire d'augmenter le nombre de backtracks lorsqu'on cherche des overlaps avec des k-mers plus courts car sinon, le temps d'exécution est bien trop important (jusquà 20min pour un seul LR)

— Trop baisser la taille min de l'overlap => Mauvaise qualité => 40 semble pas mal, peut être augmenter encore un peu

2.116 Jours 162 - 170 (Vacaces)

— Retirer la liste des k-mers déjà visités => Plus mauvais résultats

— Test sur un jeu ADP1 pour déterminer la valeur optimale de k, et la valeur optimale du seuil overlap min

— On couvre visiblement mieux le génome de référence en choisissant des k-mers plus courts, mais pas trop, bien que la qualité baisse un peu (0,1 %)

— Ne pas autoriser un trop petit overlap min par rapport à la taille des k-mers initiaux semble améliorer les résultats (moins de fragments, meilleure couverture)

— Valeur idéale de k semble être 45

— Valeur idéale d'overlap min semble être 40 (pas de grands changements avec des valeurs voisines)

— Test correction + assemblage Coli avec k = 56 => Un peu mieux qu'avec k = 64

— Test correction + assemblage Coli avec k = 42 => 7 contigs

— Test correction + assemblage Coli avec k = 45 =>

— Quand on produit un LR fragmenté, on étend à droite le dernier fragment, mais à gauche aussi ?

— La liste des k-mers visités est mal gérée dans le backtracking (quand on explore une mauvaise branche, on retire le premier des k-mers erronés de la liste, mais pas les suivants, ajoutés par l'appel récursif)

2.117 Jour 171

— Test modification de la gestion de la liste des k-mers visités lors du backtracking => Ne change rien

— Test RÀZ liste k-mers visités entre chaque linking de seeds => Ne change rien

— Test tuning des paramètres de Canu pour assemblage Coli

— Passer l'overlap nécessaire pour fusionner 2 seeds à 45 semble également améliorer les résultats

— Exposé Génoscope

2.118 Jour 172

— Cleanup méthode correction, paramètres à donner à l'exécution et non à la compilation

— Script pour faire les 5 étapes d'un coup

— Parallélisation de la méthode

— Commande : ./fullTest.sh M12D.fa ADP1.fq 45 8 63 fres.fa

— Tests de fonctionnement avec la parallélisation

2.119 Jour 173

— Tests d'assemblage / tuning des paramètres pour Ecoli

— Tests d'assemblage Coli sans les LR fragmentés => On est plus efficaces avec

— Retravail méthode correction => Livré à FX

— Test en modifiant le paramètre "repeat" de PgSA => Ne change rien, car non utilisé dans le code

— Test sans corriger les SR => Résultats pourris

— Test de défragmentage en fonction du résultat produit so far => Semble marcher, mais seg fault

— Recherche et correction de la segfault => Quand on skippait la première source, on faisait un realloc en donnant en param la longueur d'une chaine nulle

— Résultat du défragmentage : Permet de corriger autant de LR, et d'avoir une plus grande taille cumulée / taille moyenne

— Test du défrag sur tout ADP1 : Visiblement, provoque qq segfault ? => Car filtration des alignements non rendue générique...

— Correction de la filtration des alignements + lancement correction tout ADP1

2.120 Jour 174

— Découverte d'une erreur : On ne mettait pas bestn à 30 dans le script de correction

— Comparaison version frag + nofrag (temps, qualité, assemblage) => Sur M12D, on y gagne beaucoup au prix d'une légère perte de qualité (0,06 %) et de 10sec de + de runtime Sur tout ADP1,

— Recherche seuil backtrack idéal si on utilise la méthode défrag => On peut descendre au moins jusque 1 125

— Recherche valeur de k idéale si on utilise la méthode défrag

— Test déplacement incr nbBackTrack dans le code, pour être plus correct => Ne change rien, mais semble plus correct

— Test en n'augmentant pas le nombre de backtracks quand on décrémente la taille du chevauchement entre les k-mers => Beaucoup trop long (probablement car on reviste des k-mers déjà visités, d'où point du dessous)

— Test en ne ràz-ant pas la liste des k-mers visités lors d'un backtrack => Ne change rien

— => Vraiment besoin d'augmenter le nombre de backtracks quand on décrémente la taille du chevauchement

— Test en ràzant la liste des k-mers visités lorsqu'on décrémente la taille du chevauchement => Plus mauvais résultats

2.121 Jour 175 - 177

— Réunion w/ TL & AL

— Encore des erreurs de segmentation dans la production de LR non fragmentés => D'où ? ? ?

— Comparaison qualité LR fragmentés / non fragmentés => Les non fragmentés sont un peu moins précis, mais la perte est faible

— Comparaison assemblage LR fragmentés / non fragmentés sur ADP1 => On ne produit pas un seul contig avec les LR non fragmentés avec paramètres par défaut, mais on s'en approche beaucoup (deux très courts contigs en +), et couverture toujours a 99,95

— Implémentation nb max de seeds skippés

— Implémentation extension par k-mer le + fréquent en 1er => Impossible, car on indexe les k-mers et non les reads

— => Possible de simuler ça avec une table de hash / whatever ?

— Légère retouche article

— Lancement test correction sur ADP1 avec la version modifiée (1125 ol + 10 skips seeds max) du linker => 21h à tout corriger, seulement 230k de bases en moins, qualité / avLen similaires => Super intéressant

— Test assemblage ADP1 avec version modifiée linker => Résultats très similaires à la version où on teste tous les reliages possibles

— Test assemblage ADP1 en ajustant les paramètres => Un seul contig, mais toujours pas 100%

— Recherche valeur de k idéale pour l'assemblage non fragmenté => Aucun pic observé

— Lancement correction Coli avec k = 64, car pas de paramètre vraiment meilleur

2.122 Jour 178

— Suite de la recherche de valeur de k idéale pour l'assemblage non fragmenté => Aucune valeur ne se dégage vraiment

— Retouche article (TODO : CITER PARALLEL)

— Installation et tests d'autres outils de correction (Colormap et Jabba)
=> Jabba est extrêmement rapide, corrige à peu près autant de reads que nous, avec une bonne identité, mais produit des assemblages de très mauvaise qualité => Colormap est 3 fois plus rapide que nous, mais produit une correction de mauvaise qualité

— Recherche de paramètres pour bien assembler ADP1

— Assemblage Coli sans reads fragmentés, paramètres par défaut => On est meilleur que Jabba, proche du contig unique

— Lancement correction Yeast non fragmenté

2.123 Jour 179

— Suite de la recherche de valeur de k idéale pour l'assemblage non fragmenté

— Recherche de paramètres pour bien assembler ADP1 / Coli

— Optimisation code linking seeds + correction d'un bug post-optimisation

— Correction du bout de code causant probablement la segfault dans le traitement des seeds (lors du calcul de l'overlap entre deux seeds qui se suivent)

— La version optimisée est extrêmement lente sur certains LR, wtf ? => Car j'avais retiré le nb max de backtracks...

— La version optimisée fonctionne et produit les mêmes résultats

— Retouche et annotation de l'article

— Demander en réunion : Besoin de préciser error correction à chaque fois ?

— TODO : Répétition pour notre méthode dans intro / son chapitre

2.124 Jour 180

— Test passage du minOverlap à KLEN - 10 => Produit de plus mauvais résultats

— Test en mode full DBG => Semble donner meilleure avLen, meilleure cumSize, meilleure cov, plus faible identité => Problématique ? => Vient du fait que quand on arrive au bout d'un chemin pour lequel on a plus de chevaucehment de k-1, on poursuit sur ce chemin avec des chevauchements de k-2, au lieu de backtrack vers les autres chemins qui ont des chevauchements de k-1

— Grosse refonte article (répétitions, passage en mode DBG / OL graph)

— Test valeur de k idéale sur 500 LR Ecoli => Ne s'accorde pas avec la valeur trouvée pour ADP1 (91 ou 50)

— Recherche d'un consensus de valeur idéale ADP1 / Coli => Valeur située entre 55 et 60

— Trouvé une erreur dans le code ? => Ne change rien sur le jeu testé, mais pouvait être problématique dans certains cas

— Lancement full correction Coli avec k = 55, semblant être optimal

— Test assemblage sans l'étape de correction, à la main => Mêmes résultats que quand on corrige aussi

2.125 Jour 181

— Fin recherche consensus valeur idéale de k => k = 54 ou 55 sur ADP1 / Coli

— Recherche valeur idéale sur 500 LR Yeast => par consensus, 55

— Retouche article (résultats colormap / jabba, conclusion, paragraphe paramètres)

— Colormap corrige très mal, et Jabba corrige bien, mais couvre peu le génome de référence

— Idée : Si après le nombre donné de skips, il reste des seeds => Les relier quand même et fragmenter ?

2.126 Jour 181-183

— Étude résultats mapping Coli avec k = 55, et comparaison avec k = 64 => k = 64 a une meilleure taille cumulée, le reste est similaire

— Étude assemblage Coli avec k = 55 => Meilleur avec k = 64...

— Réunion w/ TL & AL

— Retrait des fichiers textes temporaires, et récupération des seeds à lier directement dans un tableau, afin de pouvoir fragmenter et forcer le reliage

— Test de l'optimisation précédente => Bug => Recherche et correction du bug
— Implémentation fusion skipping + fragmentation => Améliore encore les résultats par rapport à la version précédente
— Recherche nombre de skips idéal => 5 semble être pas mal
— Nouvelle recherche taille k idéale => Résultats semblent cohérents avec ceux observés précédemment au niveau des variations de valeurs, difficile de trouver mieux que 64, 55 diminue trop la qualité même s'il augmente la couverture, la taille totale, et la taille moyenne, et 76 diminue trop la couverture, la qualité, et la taille totale, même s'il augmente la taille moyenne
— Lancement correction full Ecoli k = 64, avec la nouvelle méthode => Seulement 2 contigs !
— Recherche paramètres pour mieux assembler
— Test full Coli avec k = 75 => L'assemblage produit est mauvais, quels que soient les paramètres de Canu
— Test full Coli avec k = 55 => Tué sans faire exprès...
— Retouche article, ajout du graphe + du schéma de seed skipping, fusion tableaux 1D / 2D
— Lancement NaS fast ADP1 => interminable

2.127 Jour 184

— Kill de NaS fast => Toujours rien après 3 jours de run
— Lancement Colormap sur Ecoli + récupérations des résultats
— Relancement notre méthode sur Coli, k = 55, car process tué par accident...
— Étude résultat Colormap sur ADP1 => Le OEA ne change rien, mais bouffe plus de temps
— Retouche schémas seed skippings, retouche article, reformulation, ...
— Relancement Colormap sur ADP1
— Vérification bibliographie

2.128 Jour 185

— Récupération résultats Coli k = 55, et test d'assemblage => Plus mauvais que k = 64

— Lancement test Coli k = 60

— Récupération résultats CoLoRMap ADP1

— Réunion w/ AL => Vérification article, tout est bon, plus qu'à synthétiser un peu

— Retouches article, remise en forme, synthèse, reformulation, ...

— Lancement CoLoRMap sur Yeast

2.129 Jour 186

— Comparaison Coli k = 60 et k = 64 => À paramètres égaux, k = 64 produit un meilleur assemblage

— Tests sur les paramètres de Canu pour n'obtenir qu'un seul contig Augmenter un peu l'estimation de la taille du génome => Ne change rien Trimmer avant l'assemblage, sans paramètres => Pas de changement pour ADP1, ni pour Coli Trimmer avant l'assemblage, avec paramètres => Aucun changement non plus

— Lancement test Coli k = 68 => Couvre moins que 64 quand on aligne autant de reads => Abandon avant la fin

— => Lancement test Coli k = 65

— Reprise schéma seed skippings et passage en NB

— Récupération de la couverture (en %) de Yeast par les LR avec Last, puis conversion en sam, puis depth => Les résultats (pour l'identité) du sam produits sont erronnés, mais semblent correct pour le depth (on couvre bien tout ADP1 et tout Ecoli en vérifiant avec cette technique)

— Récupération du coverage (en x) des 3 jeux de données LR => Que mettre pour Yeast ?

— Tests intensifs sur les paramètres de Canu (augmentation / baisse peu à peu de taille k-mer, error rate, et merSize)

2.130 Jour 187

— Poursuite des tests sur les paramètres de Canu => Pas concluant, 2 contigs avec Coli, k = 64 quoiqu'il arrive

— Relecture complète article et annotation des points à vérifier à la prochaine réunion

— Récupération résultats CoLoRMap sur Yeast

— Récupération runtimes Jabba 16 procs sur tous les jeux de données + résultats sur Yeast

— Lancement test Coli 63 sur machine perso

— Comparaison assemblage Coli 64 / Coli 65 => 64 est toujours meilleur...

— Lancement test Coli 64 avec 3 seeds skip max => Couvre moins

— Lancement test Coli avec 1500 backtracks

2.131 Jour 188 - 190

— Récupération Coli 63 + Comparaison assemblage Coli 64 / Coli 63 => Toujours meilleur avec 64....

— Comparaison de coverage (en x) de nos jeux de données => Le calcul en R fait un peu n'importe quoi (indique un couverture de 37 pour NaS fast sur Coli, alors que ce n'est pas le cas)

— Récupération Coli 1500 backtracks + comparaison Coli 1125 backtracks => Produit de meilleures métriques (avLen, cumSize, id un peu plus faible) mais un plus mauvais assemblage (reads chimériques avec plus de BT)

— Réunion w/ TL & AL => Relecture de tout le papier, correctifs + essayer d'étendre en dehors des templates

— Test en étendant en dehors des templates => Permet d'obtenir des SLR bien plus longs, qui s'alignent bien, et couvrent mieux

— Lancement 1125 BT sur Crihan, 1000 BT machine perso, avec extensions en dehors des templates => Les deux sont très similaires, 1125 semble un peu mieux

— Test d'assemblage avec extension en dehors des templatess => Très mauvais résultats sans trimming => Pareil avec trimming...

— Recherche de paramètres pour correctement assembler les LR étendus en dehors des templates : => Diminuer error rate : N'améliore rien => Augmenter error rate : Semble un peu mieux, mais toujours trop de contigs => Paramètres par défaut : N'améliore pas non plus => DONC => Étendre en dehors des templates = mauvaise idée ? Bizarre...

— Test assemblage SLR étendus avec 1000 BT => Mauvais résultats aussi

— Récupération temps premières étapes de Coli

— Lancement full Yeast en étendant seulement jusqu'au bout des tpl

— Encore des segfault dans le code du linker => Du au mauvais filtrage des courts alignements (on ne comptait pas len de l'alignement) => Corrigé, et recorrection de Coli 64 sur machine perso, voir si on peut avoir un meilleur assemblage => Toujours des segfaults dans Yeast => Parce qu'on a créé le fichier de seeds à partir du fichier d'alignements erroné, et que HG-CoLoR va chercher des seeds qui n'existent plus dans le nouveau fichier, car aucune segfault pour Coli

2.132 Jour 191

— Coli sur machine perso ne donne pas les mêmes résultats que Coli sur machine Crihan => Recherche d'où ça vient => Les SLR ne s'alignent pas aux mêmes endroits entre les deux versions, même s'ils sont identiques, wtf ? ? ? => Très peu de reads semblent être différents entre les deux versions => A cause du nouveau filtrage ? Semble que non... => Test sur un LR avec les deux différents types de filtres => Pas à cause du filtrage, même résultat dans les deux cas => RÉPONSE => Car le linker utilisé avant, pour générer l'ancien fullcoli64, était un peu différent, impossible de trouver où

— Tests d'assemblage de Coli sur machine perso => Catastrophique avec le jeu Coli corrigé sur machine perso => À cause des paramètres ?

— Modifications sur article (retour à version extension jusqu'au bordures des templates + qq correctifs / reformulations)

— Recalcul des coverage des LR initiaux avec samtools depth + average de colonne 3

— Lancement full correction ADP1 sur machine perso => Encore des segfaults, wtf ? => Sûrement dû à la modification de correctRead pendant l'exécution => Première chose à faire : Récupérer les LR qui n'ont pu être traités à cause de la modif, et les corriger (script déjà prêt, devrait être rapide)

— Même avec le bon ensemble de SLR et les bons paramètres, impossible de bien assembler Coli => Car le dnadiff sur machine perso de marche pas => Besoin d'assembler sur machine Crihan

— Test assemblage SLR étendus au delà des templates => Cohérent avec les résultats précédents, mauvais assemblage

— Test assemblage SLR obtenus sur machine perso => Mêmes résultats qu'avec l'ancien jeu SLR Coli

2.133 Jour 192

— Correction des LR ADP1 manquants : Seulement 2 corrigés de plus

— Recherche d'où viennent les segfault sur le petit jeu ADP1 => Car allocation impossible à un moment, si le read à étendre est déjà trop grand => Passer au malloc pour version finale ?

— Assemblages Coli / ADP1 sur machine Crihan, recherche de consensus => 3 contigs et 99,99 de cov (0.08 partout) ou 2 contigs et 99,95 de cov (0.085 partout) pour Coli => Possible d'avoir un seul contig a 99,99 pour ADP1 (0.07 partout) => Possible d'arrêter avant la fin du consensus pour obtenir le nombre de contigs, gagne du temps => Chercheur une valeur entre 0.07 et 0.085 pour avoir l'optimal pour les deux => 0.075 ne marche pas, toujours 3 contigs pour Coli, et 2 pour ADP1 => Modifier chaque paramètre séparément ?

— Ajouts résultats ADP1 dans le tableau

— Ajoute des % de LR alignés / sans erreurs dans le tableau

— Discussion sur les résultats alignements

— Restructuration papier, sur la partie PgSA

2.134 Jour 193

— Différence entre trim-assemble et assemble => Aucune, juste assemble est même légèrement meilleur

— Légères retouche de perfectionnement sur l'article

— Suite et fin de la recherche de consensus sur les paramètres de Canu pour bien assembler ADP1 / Coli => Possible d'avoir 2 contigs pour Coli avec error rate = 0.0825 et merLimits = 0.9955 => Possible d'avoir un contig pour ADP1 avec error rate = 0.0800 et merLimits = 0.9975 => Baisser les merLimits a 0.99 ne permet pas plus d'obtenir un bon consensus => Impossible de trouver un consensus, on corrige chaque jeu de données avec ses propres paramètres, en les précisant dans le papier

— Assemblage de tous les ensembles de LR ADP1 / Coli + remplissage du tableau