Université de Rouen Normandie - UFR Sciences et Techniques
Master 2 mention Bioinformatique – Parcours BIMS
2024-2026

Rapport de mission en alternance M2.2

De la construction à la simulation des pangénomes :

une approche bioinformatique intégrée via Asm4pg et MSpangepop

---

Présenté et soutenu par

Lucien PIAT

INRAE, Univ. Bordeaux, UMR BIOGECO 1202, F-33610 Cestas, France

Maître d'apprentissage :
Dr. Ludovic DUVAUX, Ingénieur de recherche

Université de Rouen Normandie - UFR Sciences et Techniques
Master 2 mention Bioinformatique – Parcours BIMS
2024-2026

Rapport de mission en alternance M2.2

De la construction à la simulation des pangénomes :

une approche bioinformatique intégrée via Asm4pg et MSpangepop

Présenté et soutenu par

Lucien PIAT

INRAE, Univ. Bordeaux, UMR BIOGECO 1202, F-33610 Cestas, France

Maître d'apprentissage :
Dr. Ludovic DUVAUX, Ingénieur de recherche

Remerciements

Je souhaite adresser mes remerciements à l'ensemble de l'UMR BioGeCo, à l'équipe E4E ainsi qu'aux différents groupes de travail en pangénomique, pour leurs retours toujours bienveillants sur mon travail et pour l'accueil chaleureux qui m'a été réservé.

Merci tout particulièrement à mes camarades, stagiaires, doctorants, postdoctorants et techniciens, avec qui j'ai partagé de nombreux moments et qui ont rendu les pauses déjeuner bien trop courtes. Merci à MG, Grég, Stheve, Maurane, Coline, Thomas, Anouck, Charlotte, Bonnie, Bérénice, Adé, Erwan, Eva, Fernanda, Julien, Laura, Lucas, Océane et Quentin.

Merci à Su et à Florent pour leurs conseils avisés tout au long de mon alternance.

Merci aux enseignants du master BIMS, dont la rigueur et l'enthousiasme ont rendu les moments passés à Rouen particulièrement enrichissants. Merci également à Hadrien, Lale, Antoine, Louison, Rayan, Lamia, Maël et Luis pour m'avoir accueilli à bras ouverts dans leur promotion de M2, sans doute la plus soudée que j'aie eu l'occasion de connaître.

Merci à Ludovic pour ta plume, ta pédagogie et ta patience. Merci de m'avoir pris sous ton aile pendant ces dix-huit mois, au cours desquels j'ai beaucoup appris. Merci également de m'avoir encouragé à présenter notre travail et de m'avoir accompagné à de nombreux congrès, qui seront, j'en suis certain, un véritable tremplin pour la suite.

Enfin, merci à Claire, qui a dû supporter mes longs tunnels d'explications sur la taille du pédoncule du chêne pédonculé.

Table des matières

Table des figures

Liste des sigles et abréviations

CI/CD : *Continuous Integration / Continuous Deployment* (Intégration continue / déploiement continu)

DAG : *Directed Acyclic Graph* (Graphe orienté acyclique)

FAIR : *Findable, Accessible, Interoperable, Reusable* (Trouvable, accessible, interopérable, réutilisable)

GED : *Graph Edit Distance* (Distance d'édition de graphe)

GFA : *Graphical Fragment Assembly*

HiFi : *High Fidelity* (Haute fidélité — méthode de séquençage PacBio)

Hi-C : *High-throughput Chromosome Conformation Capture* (Capture de conformation chromosomique à haut débit)

HPCC : *High Performance Computing Cluster* (serveurs de calcul haute performance)

INRAE : Institut national de recherche pour l'agriculture, l'alimentation et l'environnement

LAI : *LTR Assembly Index* (Indice d'assemblage des LTR)

LTR : *Long Terminal Repeat* (Longue répétition terminale)

MC : Minigraph-Cactus

MRCA : *Most Recent Common Ancestor* (Ancêtre commun le plus récent)

Mpb : Méga paires de bases soit 10^6 pb

ONT : Oxford Nanopore Technologies

POO : Programmation Orientée Objet

PVG : *Pangenome Variation Graph* (Graphe de variation pangénomique)

QC : *Quality Control* (Contrôle qualité)

SNP : *Single Nucleotide Polymorphism* (Polymorphisme nucléotidique simple)

SV : *Structural Variant* (Variant structural du génome)

TE : *Transposable Element* (Élément transposable)

UMR : Unité Mixte de Recherche

VCF : *Variant Call Format* (Format d'appel de variants)

Glossaire

Benchmarking : Évaluation comparative des performances d'outils informatiques.

Bloc de recombinaison : Région du génome où les variants génétiques sont parfaitement co-hérités. Biologiquement, cela correspond à des segments situés entre deux *crossing-overs*, de sorte que tous les sites à l'intérieur du bloc sont transmis comme une seule unité au cours de la méiose.

Bulle (de variation) : Terme générique désignant un motif de divergence et de reconvergence dans un graphe.

Coalescence : Processus stochastique modélisant l'histoire généalogique d'un échantillon d'allèles en remontant le temps jusqu'à leur MRCA.

Éléments transposables : Séquences d'ADN répétées capables de se déplacer et de se multiplier de manière autonome au sein du génome.

Haplotype : Désigne, pour un individu donné, l'une des copies constitutives de son génome (ou chromosome).

Hackathon : Événement collaboratif intensif réunissant des développeurs et experts sur une courte période pour concevoir et programmer rapidement des solutions informatiques.

Modèle démo-génétique : Modèle couplant la dynamique démographique des populations (e.g. variations de taille, migration) et les processus génétiques d'évolution des génomes (e.g. mutation, recombinaison).

Panmixie : État d'une population au sein de laquelle les individus se reproduisent de manière totalement aléatoire.

Ploïdie : Nombre d'haplotypes par cellule. Elle détermine si l'organisme est **haploïde** (une copie), **diploïde** (deux copies) ou **polyploïde** (plusieurs copies).

Règle (Snakemake) : Unité fondamentale d'un workflow Snakemake décrivant une étape de calcul, définit généralement les fichiers d'entrée, sortie et la commande nécessaire à la transformation.

Scaffold (échafaudage) : Assemblage de plusieurs contigs ordonnés et orientés le long d'un chromosome.

Snarl : Équivalent d'une bulle dans un graphe bi-orienté. Ce sous-graphe est délimité par deux sommets frontières, agissant comme un nœud source et un nœud puits, qui isolent une région de variation complexe (incluant inversions ou cycles) du reste du graphe.

Tree-sequence (Séquence d'arbres) : Structure de données efficace codant l'histoire évolutive complète d'un génome. Elle représente la succession des arbres généalogiques le long de la séquence chromosomique.

Unitigs : Un chemin maximal et linéaire dans un graphe, dont les nœuds internes ne présentent aucun embranchement (degré entrant et sortant égal à un), et qui peut être contracté en un unique nœud sans altérer la structure topologique du graphe.

Workflow : Séquence organisée et automatisée d'étapes ou d'analyses, où la sortie d'une étape sert d'entrée à une ou plusieurs étapes suivantes.

I | Introduction

1. L'Unité Mixte de Recherche Biogeco d'INRAE

L'unité mixte de recherche (UMR) 1202 BioGeCo (Biodiversité, Gènes et Communautés) rassemble environ 115 chercheurs, techniciens et étudiants sous la double tutelle de l'Université de Bordeaux et d'INRAE (Institut national de recherche pour l'agriculture, l'alimentation et l'environnement). Ses effectifs sont répartis sur deux sites : l'un en périphérie de la forêt des Landes, sur le campus forêt/bois de Cestas-Pierroton, et l'autre sur le campus de l'Université de Bordeaux à Pessac. L'unité est organisée en sept équipes partageant un objectif : étudier la biodiversité, ses fonctions, sa structure et son évolution, à différentes échelles du vivant, des gènes aux populations[1] (fig. I.1).

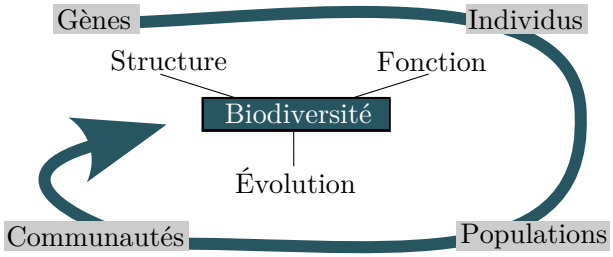

FIGURE I.1 – **Les axes de recherche à BioGeCo**

Mon équipe, E4E (Évolution dans les Écosystèmes Forestiers), se concentre sur l'étude de l'évolution des écosystèmes forestiers. Ses modèles d'études se distribuent en Europe et en Guyane française et incluent entre autres les forêts de pins maritimes (*Pinus pinaster*) d'Europe du sud, les grandes chênaies françaises et d'Europe (genre *Quercus*), les populations relictuelles d'ifs (*Taxus baccata*) distribuées dans toute l'Europe ou encore des espèces de la forêt primaire guyanaise exploitées pour leur bois (e.g. *Dicorynia sp.*). Elle est majoritairement composée de généticiens et d'écologues, parmi lesquels des pionniers de la génétique évolutive des arbres tel qu'Antoine Kremer[2], qui étudient l'évolution de la biodiversité à des échelles de temps écologiques mais également micro-évolutives plus longues (différentiation des populations et des espèces)[3].

2. Variations structurales et diversité

Comprendre les mécanismes génétiques qui sous-tendent l'adaptation des organismes à leur environnement constitue un enjeu central de la biologie évolutive. Cette compréhension repose entre autres sur l'étude de la diversité génétique, c'est-à-dire l'ensemble des différences nucléiques observées entre individus d'une même espèce ou entre espèces proches. Historiquement, cette diversité a été principalement décrite à l'aide de polymorphismes nucléotidiques simples (SNP) et de microsatellites, des marqueurs largement utilisés en génétique des populations. Toutefois, la variabilité génomique ne se limite pas

à ces mutations ponctuelles. Elle inclut également des réarrangements chromosomiques de grande ampleur, regroupés sous le terme de variations structurales (SV). Ces dernières englobent des insertions, des délétions, des inversions, des duplications ou encore des translocations de segments génomiques (fig. I.2). Longtemps restées en marge des analyses génomiques, les SV contribuent néanmoins fortement à la diversité des populations et exercent une influence majeure sur le phénotype [1], notamment lorsqu'elles affectent des gènes ou leurs régions régulatrices. Leur prise en compte ouvre ainsi des perspectives nouvelles pour l'identification des bases génomiques de l'adaptation, tout en posant des défis méthodologiques importants.

Afin de mieux appréhender cette complexité structurale, un nouveau cadre conceptuel s'est imposé ces dernières années : **la pangénomique**. Initialement développée chez les bactéries, cette approche est désormais étendue à l'ensemble du vivant. Elle vise à décrire l'intégralité des séquences génétiques présentes au sein d'une population, d'une espèce ou d'un groupe d'espèces apparentées [4], en intégrant aussi bien les SNP que les SV. Plusieurs formalismes mathématiques ont été proposés pour représenter les pangénomes, à l'instar des graphes de De Bruijn [2]. Cependant, ces modèles peinent à capturer les réarrangements de grande taille et à représenter fidèlement les régions répétées. Dans ce contexte, les graphes de variation, ou *pangenome variation graphs* (PVG), se sont imposés comme une représentation particulièrement adaptée (fig. I.2). Les nœuds y correspondent à des segments de séquence, reliés par des arêtes représentant les connexions possibles, tandis que les chemins à travers le graphe décrivent les haplotypes individuels. Cette structure permet de distinguer les régions communes à l'ensemble des individus et des variants propres à certains haplotypes, facilitant ainsi la détection et l'étude des variations structurales [3].

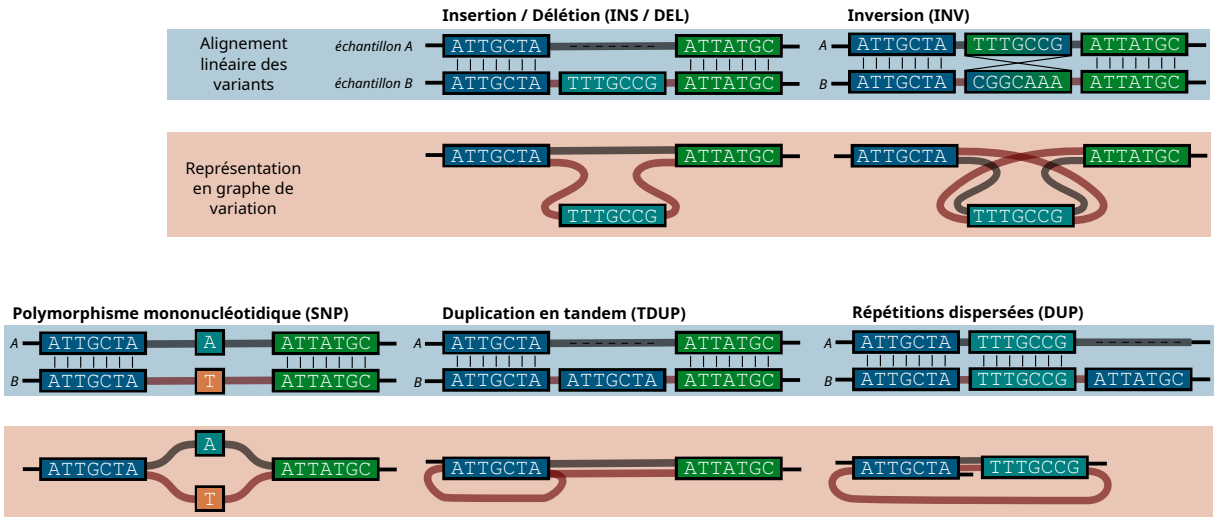

FIGURE I.2 – **Principaux types de variations structurales génomiques**. Deux haplotypes (A et B) sont représentés soit par un alignement linéaire (panneaux supérieurs bleus) soit sous forme d'un graphe de variation (panneaux inférieurs rouges). Les blocs de même couleur représentent des séquences homologues. Les chemins de chaque haplotype sont définis par la succession d'arêtes d'une couleur donnée.

La construction de PVG fiables repose toutefois sur une condition préalable essentielle : la disponibilité d'assemblages génomiques de haute qualité. En raison de leur taille et de leur complexité, les SV ne peuvent être correctement identifiées qu'à partir de données capables de couvrir intégralement les régions concernées. L'essor récent de la pangénomique eucaryote est donc étroitement lié à l'avènement des technologies de séquençage à lectures longues. Les approches à lectures courtes (<300 pb), longtemps privilégiées pour

leur faible coût, montrent leurs limites dans l'analyse des régions répétées ou réarrangées. À l'inverse, les plateformes PacBio HiFi et Oxford Nanopore Technologies (ONT) produisent des lectures d'environ 15 000 pb, et dans les cas extrêmes des millions de bases [4]. Ces avancées technologiques ont nécessité un développement d'algorithmes d'assemblage performants, capables de produire des génomes de haute qualité de manière routinière et, dans certains cas, de séparer les haplotypes parentaux, c'est-à-dire les deux copies du génome héritées respectivement du père et de la mère [5]. Ces progrès sont particulièrement déterminants chez les plantes, dont les génomes, riches en éléments transposables (TE) et en variations structurales, sont souvent plus labiles que ceux des vertébrés [6]. **Toutefois, la mise en œuvre de ces algorithmes reste complexe, car elle implique la combinaison de multiples types de données, rendant le recours à des workflows dédiés indispensable.**

Grâce à la plus grande disponibilité d'assemblages de haute qualité, des outils tels que `PGGB` (*PanGenome Graph Builder*) [7] ou minigraph-cactus [8] permettent aujourd'hui de générer des PVG (fig. I.3, A–B). Néanmoins, le recul vis-à-vis de ces méthodes récentes demeure limité. Aucun cadre de *benchmarking* standardisé ne permet actuellement de comparer la qualité des graphes obtenus. De plus, l'absence de simulateurs pour générer des pangénomes synthétiques réalistes empêche une évaluation objective des performances de ces méthodes. **Ce manque de validation rigoureuse constitue un verrou méthodologique majeur, auquel s'ajoutent des enjeux de reproductibilité, appelant au développement de *workflows* standardisés et à l'adoption de bonnes pratiques telles que la conteneurisation des outils bioinformatiques.**

C'est dans ce contexte méthodologique et conceptuel que s'inscrit le projet Pangenoak, coordonné par mon maître d'apprentissage Ludovic Duvaux. Ce projet vise à exploiter la pangénomique pour caractériser les variations structurales chez les chênes blancs européens, notamment le chêne pédonculé (*Quercus robur*) et le chêne sessile (*Quercus petraea*). Ces espèces appartiennent à un complexe d'une quinzaine d'espèces interfertiles, dont les génomes au fort polymorphisme partagé, compliquent l'identification de marqueurs génétiques discriminants (mais voir [9]). L'étude des SV constitue ainsi une approche complémentaire prometteuse permettant de capturer des réarrangements de grande ampleur susceptibles d'améliorer la résolution phylogénétique, voire de porter des gènes adaptatifs (fig. I.3, B). La détection de gènes potentiellement adaptatifs, notamment aux environnements chauds et secs, revêt une importance particulière dans un contexte de changement climatique, alors que le dépérissement des chênes est déjà observé en Europe [10].

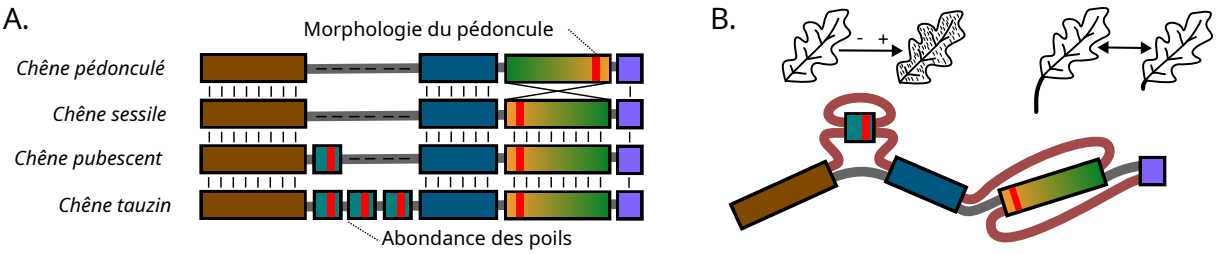

FIGURE I.3 – **Exemple fictif d'utilisation d'un PVG :** À partir de l'alignement de quatre espèces de chênes (A), le PVG (B) met en évidence deux variants structuraux. Une variation du nombre de copies (nœud turquoise), liée à des éléments transposables, serait associée à un phénotype (e.g. l'abondance des poils chez le chêne tauzin), tandis qu'une inversion chromosomique (jaune–vert) pourrait affecter la morphologie du pédoncule (chêne pédonculé).

3. Objectifs de mon travail

Mes travaux ont été réalisés dans le cadre du projet Pangenoak. Mon alternance avait pour objectif de concevoir et de mettre en place des outils bioinformatiques adaptés à la construction et à l'analyse de pangénomes, en réponse aux limites identifiées dans les approches existantes. Ce travail s'est organisé autour de deux axes complémentaires : (i) la production d'assemblages génomiques de haute qualité et (ii) le développement d'outils de simulation innovants (fig. I.4).

Le premier axe portait sur la production d'assemblages génomiques de qualité chromosomique, indispensable à l'étude des variations structurales. Le passage à l'échelle sur de larges cohortes requiert de réduire toute intervention manuelle au strict minimum et nécessite des *workflows* automatisés et reproductibles. Dans ce cadre, j'ai assuré la maintenance et l'évolution du programme `asm4pg`, afin d'en faire un outil de production robuste, intégrant les avancées algorithmiques récentes et garantissant sa portabilité par conteneurisation.

Le second axe, qui constitue le cœur de mon travail, visait à répondre à un verrou méthodologique majeur du domaine : l'absence de référentiel permettant d'évaluer objectivement les outils de construction de pangénomes. Pour y remédier, nous avons conçu un simulateur, `MSpangepop`, que j'ai ré-implanté entièrement. Le *workflow* est capable de générer des pangénomes synthétiques biologiquement réalistes dont la structure est entièrement maîtrisée. Pour cela, il couple (i) la création, par des modèles de coalescence, de généalogies réalistes sous des scénarios démo-génétiques avec (ii) un algorithme dédié à la génération de PVG que j'ai développé *de novo*. `MSpangepop` permet ainsi d'établir une base de référence contrôlée pour le *benchmarking* d'outils de pangénomiques les plus avancés.

Ces développements s'inscrivent dans une démarche de science ouverte et respectent strictement les principes FAIR (*Findable, Accessible, Interoperable, Reusable*). En fournissant à la communauté pangénomique des outils reproductibles et des cadres d'évaluation contrôlés, ce travail pose les bases d'analyses pangénomiques plus fiables et ouvre la voie à une exploitation rigoureuse des variations structurales à grande échelle.

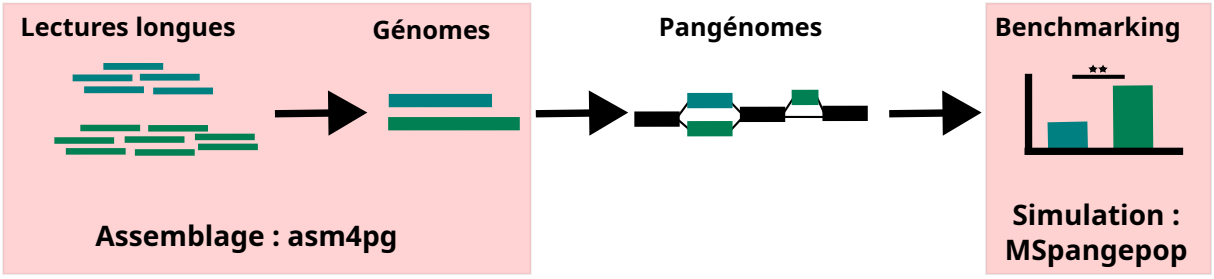

FIGURE I.4 – **Vue d'ensemble des objectifs de mon apprentissage.** Le schéma illustre le flux de traitement de données en pangénomique eucaryote. Les cadres rouges matérialisent mes deux axes de développement : 1) L'optimisation du *workflow* d'assemblage ; 2) La mise en place d'un outil de création d'une base de référence contrôlée pour le *benchmarking* des outils de pangénomique.

II | Principes, méthodes et données des développements

1. Environnement informatique

Durant mon alternance, j'ai utilisé un poste de travail sous `Ubuntu (GNU/Linux)` pour me connecter à deux clusters de calcul essentiels (HPCC, *High Performance Computing Cluster*). Le premier, le `CBiB` [5] (Centre de Bioinformatique de Bordeaux), m'a principalement servi pour le stockage de jeux de données volumineux, grâce à ses quotas larges. Le second, le cluster `Genotoul-Bioinfo` basé à Toulouse, est une ferme de calcul massive (environ 5 000 cœurs, 83 To de RAM et 7,5 Po d'espace disque [6]). Je l'ai privilégié pour les tâches intensives ; sa puissance est idéale pour des calculs à grande échelle et il est de plus très utilisé par la communauté pangénomique. Sa version récente de `SLURM` (*Simple Linux Utility for Resource Management*) offre également une meilleure compatibilité avec mes outils. Cet ordonnanceur m'a permis d'exécuter des calculs en parallèle sur l'infrastructure. J'ai utilisé aussi sur `Genotoul-Bioinfo` une instance de `code-server` [7], un environnement de développement intégré basé sur `VScode`, auquel j'accède depuis un navigateur, me permettant de travailler efficacement malgré les ressources limitées de mon poste local.

2. Pratique professionnelle

A) Veille bibliographique et technologique

Ma veille scientifique et technologique reposait sur une combinaison de pratiques complémentaires. Les échanges directs avec la communauté constituaient ma principale source d'information, à travers les réunions des réseaux pangénomiques et ma participation à un *journal club* régulier à INRAE, qui me permettaient de suivre les avancées du domaine. En parallèle, je portais une attention particulière aux outils et ressources développés par la communauté, notamment via la consultation et la contribution à la page GitHub `awesome-pangenomes` [11]. Cette veille était complétée par des médias de vulgarisation tels que la newsletter Jebif [8] ou le subreddit r/bioinformatics. Ces informations étaient ensuite organisées au moyen d'outils de recherche et de suivi bibliographique. J'utilisais *Google Scholar* pour les recherches par mots-clés et ResearchRabbit [9] pour centraliser les articles, que je consultais régulièrement afin d'identifier de nouvelles publications en lien avec la pangénomique, les variants structuraux et l'outil `msprime`.

B) Bonnes pratiques de programmation et développement logiciel

Dans un contexte où les outils sont destinés à des biologistes et seront maintenus par d'autres développeurs, j'ai adopté une démarche qualité alignée sur les principes **FAIR**

(*Findable, Accessible, Interoperable, Reusable*). Pour mes développements en `Python`, j'ai privilégié une architecture en **Programmation Orientée Objet (POO)**, garantissant une forte modularité et facilitant la réutilisation du code. L'écriture est réalisée via l'environnement `code-server`, qui me permet d'intégrer l'analyse statique et la vérification de type pour prévenir les erreurs en amont de l'exécution. L'intégralité du projet est versionnée sous `Git`, assurant une traçabilité complète des modifications et une gestion collaborative efficace. Afin de garantir la robustesse et la reproductibilité des analyses, j'ai mis en place des scripts d'intégration et déploiement continu (CI/CD). Hébergé sur la `ForgeINRAE` [10], ces processus exécutent automatiquement des tests et vérifient les dépendances à chaque mise à jour (*push*). En parallèle, pour assurer la portabilité de l'outil entre machines, j'ai recours à la **conteneurisation**. Cette encapsulation de l'environnement logiciel fige les versions des outils et garantit une exécution identique quel que soit l'infrastructure sous-jacente. En complément de ces outils, et à des fins de documentation et d'optimisation, le modèle de langage `Claude Opus 4.5` a été ponctuellement utilisé pour la relecture critique et l'amélioration de certains scripts.

Enfin, une attention particulière a été portée à l'utilisabilité pour des non-informaticiens. L'ergonomie des scripts a été pensée pour limiter le nombre de paramètres exposés, guidant ainsi l'utilisateur dans la configuration de l'outil. J'ai également implanté un système de **journalisation** verbeux et explicite. Conçu pour être lisible par un biologiste, ce système permet de suivre les étapes de l'analyse en temps réel et, en cas d'échec, de diagnostiquer l'origine du problème sans nécessiter une intervention technique immédiate.

C) Communication des travaux

La communication autour du projet reposait en grande partie sur des **échanges oraux majoritairement planifiés, intégrés à la démarche méthodologique du projet**. Ces sessions prenaient place dans différents contextes : discussion informelles, réunions de projet, réunions d'équipe E4E, présentation du projet lors de séminaires, ateliers de réseaux thématiques en pangénomique, congrès tels que les JOBIM, Journées Ouvertes en Biologie, Informatique et Mathématiques [11], ainsi que lors de journées *« hackathon »* dédiées au projet Pangenoak. Ces journées, impliquant mon encadrant et le doctorant du projet Pangenoak, étaient totalement consacrées au projet et constituaient des moments clés de travail collectif, centrées sur des discussions approfondies et concrètes autour du code et des orientations conceptuelles. Les supports de présentation et documents étaient élaborés de manière collaborative à l'aide de `Google Slides` et `Overleaf`, facilitant les échanges et les révisions. À l'issue de ces grandes phases de discussion, nous définissions une liste de tâches structurée, correspondant aux objectifs à atteindre pour l'itération suivante. J'en estimais le temps de réalisation, puis engageais une phase de développement et de tests. Les résultats obtenus étaient synthétisés dans un compte-rendu rédigé en LaTeX, servant de base à l'analyse collective et à l'ajustement des objectifs. Ce fonctionnement s'inscrivait dans un **cycle de travail itératif**, particulièrement adapté au développement d'outils nouveaux, qui nécessite une réflexion continue plutôt qu'un plan strictement préétabli. Enfin, le **suivi asynchrone** du projet était assuré via la `ForgeINRAE`. L'accessibilité permanente du code permettait à mon encadrant et aux collaborateurs de suivre l'avancement, de proposer des améliorations ou des corrections sous forme d'*issues*, et d'assurer un suivi précis de la résolution des problèmes tout au long du développement.

3. Méthodologie et Outils Communs

Le développement des deux outils sur lesquels je me suis concentré, `Asm4pg` [12] et `MSpangepop` [13], est basé sur un ensemble de méthodes communes décrites ci-après.

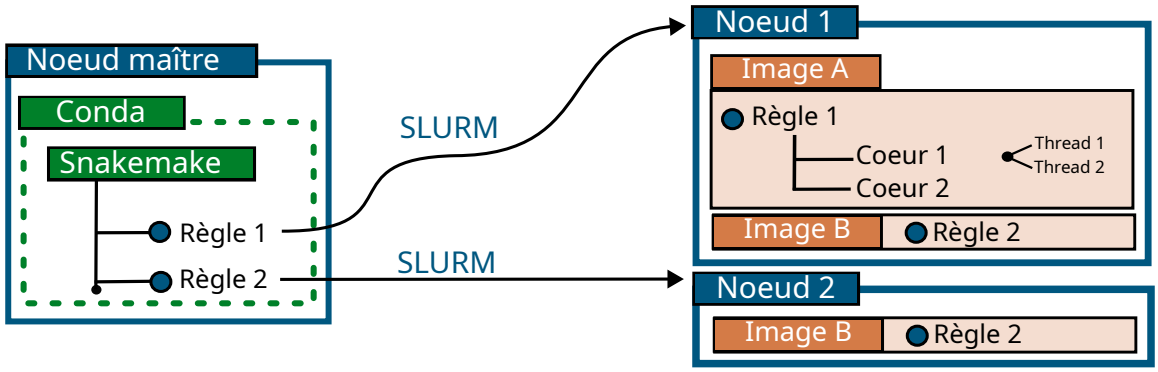

FIGURE II.1 – **Architecture de conteneurisation et de parallélisation des *workflows*** : `Snakemake`, installé dans un environnement `Conda`, orchestre l'exécution des scripts `Python` grâce au règles en utilisant des images `Apptainer` spécifiques (Image A, Image B). La parallélisation s'opère à trois niveaux : multi-nœuds via `SLURM` (Nœud 1, Nœud 2), multi-coeur au sein d'un nœud (Cœur 1, Cœur 2), et multi-exétron [14] (*thread*) au sein d'un cœur (Thread 1, Thread 2).

A) Langages et paradigmes de programmation

Le cœur de mes développements a été réalisé en `Python` (v. 3.11.13), qui constituait l'outil principal de ce travail. En tant que langage **haut niveau**, Python se distingue par une syntaxe claire et un écosystème de bibliothèques particulièrement riche, le rendant très adapté au prototypage rapide en contexte de recherche. Il m'a permis de concevoir, tester et faire évoluer rapidement de nouvelles fonctionnalités avant d'envisager d'éventuelles optimisations. Un atout majeur de Python, dans le cadre de ce projet, réside dans son support natif de la POO. Cette approche, fondée sur la structuration du code en « objets » combinant données (attributs) et comportements (méthodes), m'a permis de définir des abstractions logiques claires. Il en résulte un code plus **modulaire**, plus **maintenable** et **réutilisable**, facilitant l'évolution des outils développés.

En complément, le langage `Bash` a été utilisé comme matrice reliant l'ensemble des scripts et les infrastructures HPCC, `Bash` permet d'automatiser les tâches en ligne de commande, d'orchestrer les différents scripts Python et de piloter les *workflows*, notamment pour la soumission et le suivi des calculs.

B) Orchestration des *workflows*

Pour lier ces scripts en *workflows* robustes, j'ai utilisé `Snakemake` [12] (v. 8.4.7). Il s'agit d'un système de gestion de flux de travail (*workflow*) basé sur `Python`, très utilisé en bioinformatique car il permet de définir des chaînes d'analyse complexes de manière lisible et reproductible. Dans cette architecture, `Snakemake` invoque les différents scripts `Python`. Les paramètres sont transmis à ces scripts de manière standardisée grâce à la librairie `argparse`, qui permet de *parser* (analyser) et de valider les arguments passés en ligne de commande. L'utilisation de `Snakemake` offre des avantages de suivi cruciaux : il

capture automatiquement les sorties standard et d'erreur de chaque **règle** (étapes, voir glossaire) dans des fichiers journaux dédiés, assurant une traçabilité complète. Il permet également d'évaluer la consommation des ressources, telles que la mémoire vive et le temps CPU, pour chaque étape. L'atout majeur de `Snakemake` est sa gestion native de la **parallélisation** à plusieurs niveaux (fig. II.1). Les règles du *workflow* sont exécutées en fonction de leurs dépendances, mais `Snakemake` peut lancer des tâches indépendantes en parallèle. Grâce à son intégration avec `SLURM`, il peut distribuer ces tâches sur différents nœuds de calcul du cluster (*multi-cœur*). Au sein d'un même nœud, il peut allouer plusieurs cœurs de calculs à un outil ou exploiter le *multi-exétron* (*multi-threading*) de `Python` pour accélérer les calculs, maximisant ainsi l'efficacité.

C) Conteneurisation et reproductibilité

Pour garantir la reproductibilité des analyses et la portabilité entre machines, l'ensemble des *workflows* a été conteneurisé. J'ai utilisé `Apptainer` [13], un outil de conteneurisation conçu pour les HPCC, qui encapsule un outil et toutes ses dépendances (librairies, versions) dans un seul fichier image immuable. J'ai adopté une utilisation granulaire : plutôt qu'une seule image monolithique, j'ai créé des images `Apptainer` distinctes pour différents groupes d'outils. Cette séparation facilite grandement la maintenance et la mise à jour des dépendances. `Snakemake` s'intègre parfaitement à cette approche : il est possible de spécifier dans le *workflow* une image différente pour chaque règle (fig. II.1). Lors de l'exécution, `Snakemake` télécharge l'image requise depuis un registre (dans notre cas, la `ForgeINRAE`) et l'active automatiquement. Ce système est couplé à la `CI/CD` : toute mise à jour de la recette de construction d'une image sur `Git` déclenche une mise à jour du registre pour les exécutions futures. Cette stratégie de conteneurisation est en outre complétée par une seconde couche permettant de figer la version de `Snakemake`, l'outil étant lui-même inclus dans un environnement `Conda` dédié. L'ensemble des dépendances requises sur la machine hôte est ainsi réduit au strict minimum : `Apptainer` et `Conda`.

4. Ressources pour les assemblages

`asm4pg` est un *workflow* d'assemblage ***de novo*** de génomes, conçu pour traiter les données de séquençage à **lectures longues**. Son ambition première est de standardiser et de simplifier la production de génomes de haute qualité, proches en contiguïté d'un génome de référence, tout en minimisant l'intervention de l'utilisateur. L'objectif est de permettre à un biologiste de lancer, via une seule commande, l'assemblage complet de nombreux échantillons, une fonctionnalité indispensable aux projets de pangénomique.

La version initiale du *workflow* [15], bien que fonctionnel, reposait sur un écosystème logiciel datant de novembre 2022. L'ajout itératif de nouvelles fonctionnalités, avait entraîné une complexité accrue du code. Cette complexité se manifestait par une duplication de code significative entre les différentes modalités d'exécution. Une **refactorisation** était donc nécessaire pour améliorer la robustesse, la lisibilité du *workflow* et pour y intégrer les avancées méthodologiques les plus récentes.

A-Asm4pg) Nouveaux outils et fonctionnalités

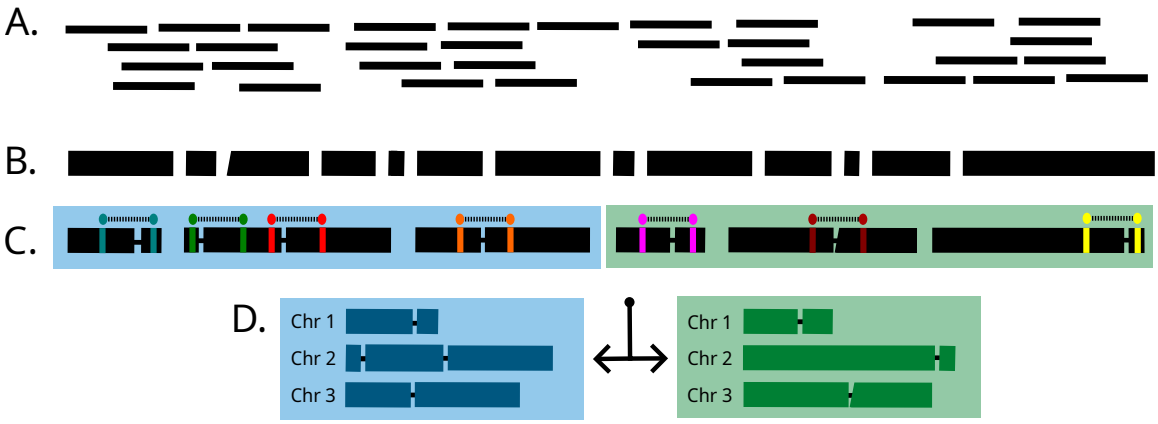

FIGURE II.2 – **Étapes clés d'un assemblage *de novo*** : **(A)** Les lectures longues (ex : PacBio HiFi) sont assemblées en **(B) contigs** (séquences assemblées contiguës). **(C)** Ces contigs sont ensuite phasés par haplotypes, ordonnés et orientés à l'aide de données de proximité **Hi-C** pour former des *scaffolds* (échafaudages). **(D)** Finalement, les *scaffolds* sont ordonnés et ancrés en pseudo-chromosomes (par ex. avec `RagTag`).

Depuis 2022, le paysage des outils d'assemblage a été profondément renouvelé. L'assembleur `Hifiasm` [5] s'est imposé comme l'outil de référence pour les données PacBio HiFi. Il repose sur une approche par **graphes d'assemblage phasés** et excelle dans la production d'assemblages à haute contiguïté. Sa capacité la plus remarquable est de résoudre nativement les haplotypes, du génome chez les organismes diploïdes, un processus nommé **phasage** (fig. II.2, D). Les versions récentes (nous utilisons la `0.24.0-r703`) ont non seulement réduit drastiquement les temps de calcul, mais ont aussi amélioré le mode Hi-C, qui utilise les données de capture de conformation des chromosomes pour affiner le phasage et la contiguïté (fig. II.2, C). Finalement, un nouveau module pour l'assemblage des données ONT est apparu étendant encore plus les cas d'usages.

Dans cette même logique, l'outil `YaHS` (*Yet another Hi-C Scaffolder*) a été intégré. `YaHS` est un outil dédié à l'**échafaudage** (***scaffolding***). Sa fonction est d'associer les contigs produits par `Hifiasm` en utilisant les signaux de proximité issus des données Hi-C, permettant ainsi la reconstruction de **pseudo-chromosomes** (fig. II.2, D). Pour les cas où un génome de référence de haute qualité existe, `RagTag` [14] a été ajouté au *workflow*. Cet outil réalise un *scaffolding* guidé par référence : il ordonne, oriente et raboute les contigs de l'assemblage en les alignant sur la référence sans pour autant modifier les SV présentes au sein des contigs.

Enfin, l'étape de **contrôle qualité (QC)** a été considérablement enrichie. `QUAST` (*Quality Assessment Tool for Genome Assemblies*) un outil très utilisé par la communauté est maintenant intégré au *workflow* et permet d'avoir une vision globale de la qualité de plusieurs assemblages. J'ai également ajouté `LTR_Finder`, un outil permettant d'évaluer la qualité des assemblages en calculant le **LAI** (*LTR Assembly Index*), un indicateur robuste de la complétude des régions intergéniques complexes basé sur l'identification des **rétrotransposons à longues répétitions terminales (LTR)**.

Outre ces intégrations principales, l'ensemble des 12 nouveaux outils ajoutés au *workflow* peut être consulté sur le répertoire d'asm4pg (voir ForgeINRAE)

B-Asm4pg) Jeux de données de validation

Pour mesurer les performances des nouvelles versions d'`asm4pg`, un jeu de données public de haute qualité a été utilisé. Il s'agit des données produites pour le chêne pédonculé (*Quercus robur*) par le projet **Darwin Tree of Life (DTOL)** [15]. Ce jeu de données de référence inclut à la fois des lectures PacBio HiFi et des données Hi-C, constituant un *benchmark* idéal. L'outil a ensuite été utilisé sur un jeu de données plus large, généré par l'UMR, comprenant 26 autres haplotypes de diverses espèces de chênes blancs européens. À des fins de tests de robustesse et de généricité, des assemblages ont également été réalisés sur des espèces taxonomiquement éloignées (tab II.1).

TABLE II.1 – Jeux de données utilisés pour la validation technique d'`asm4pg`

Organisme	**Projet/Échantillon**	**Taille génome**
Quercus robur	dhQueRobu3.1[18] (DTOL [15])	~820 Mpb
Homo sapiens	HG002 (GIAB[19])	~3.1 Gpb
Adalia bipunctata	PRJEB45127 [16] (DTOL [15])	~475 Mpb
Ganoderma boninense	Données non publiées	~43 Mpb

5. Simulation de pangénomes

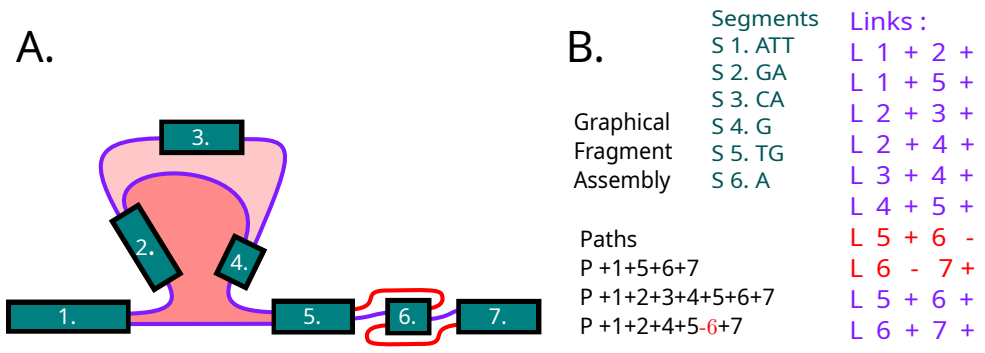

FIGURE II.3 – **Un variant emboîté représenté au format GFA. (A)** Représentation d'un variant emboîté par un PVG : une insertion (nœud 3) est incluse dans une insertion (nœuds 2 et 4) préalablement présente entre les nœuds *core* 1 et 5. On remarque aussi une inversion, figurée par les liens rouges, au niveau du nœud 6. **(B)** Représentation du PVG au format GFA à l'aide de trois types d'enregistrements : les **S**egments décrivant les nœuds du graphe ; les **L**iens (Links), définissant la topologie et les connexions possibles entre segments ; ainsi que des chemins (**P**aths), correspondant à des parcours particuliers du graphe. Les +/- définissent le sens de lecture du nœud [16].

Le second volet de mon travail a porté sur le *workflow* `MSpangepop` [13], un outil conçu pour la simulation de pangénomes en contexte de génétique des populations. À mon arrivée, `MSpangepop` était à l'état de prototype. Le flux de travail initial reposait sur trois étapes distinctes. Premièrement, il s'agissait de simuler des individus au sein de populations. Deuxièmement, un fichier VCF (*Variant Call Format*, listant des variations dans les génomes) était généré pour décrire les SV de ces individus. Troisièmement, ce VCF était utilisé comme entrée pour les outils de construction pangénomiques standards, afin de le convertir en un graphe de variation. Cette approche a rapidement montré ses limites.

D'une part, le format VCF, en rapportant systématiquement les variations à un génome de référence linéaire, s'est révélé structurellement inadapté pour représenter fidèlement la complexité des génomes, notamment les variants **emboîtés** (un variant survenant à l'intérieur d'un autre, fig. II.3.A). D'autre part, cette méthode nous rendait dépendants de ces mêmes outils de construction de graphes dont nous souhaitions, par ailleurs, évaluer les performances. Leurs biais et leurs limites étant encore mal caractérisés, il était impossible de savoir si les erreurs provenaient de la simulation ou de l'outil de construction. Pour surmonter ces écueils, il est apparu nécessaire de s'abstraire totalement du format VCF et des outils de construction intermédiaires. L'objectif a donc été redéfini : développer une méthode *de novo* capable de convertir **directement** un ensemble de séquences génétiques **simulées** en populations, incluant des SV, en un pangénome complet et « topologiquement exact », lui-même représenté au format GFA [16] (*Graphical Fragment Assembly*, fig. II.3.B). Cette approche, pour laquelle aucune méthode n'existait à notre connaissance, a nécessité un travail exploratoire pour développer et valider l'algorithme de conversion.

A-MSpangepop) Simulation de données génétiques populationnelles par coalescence

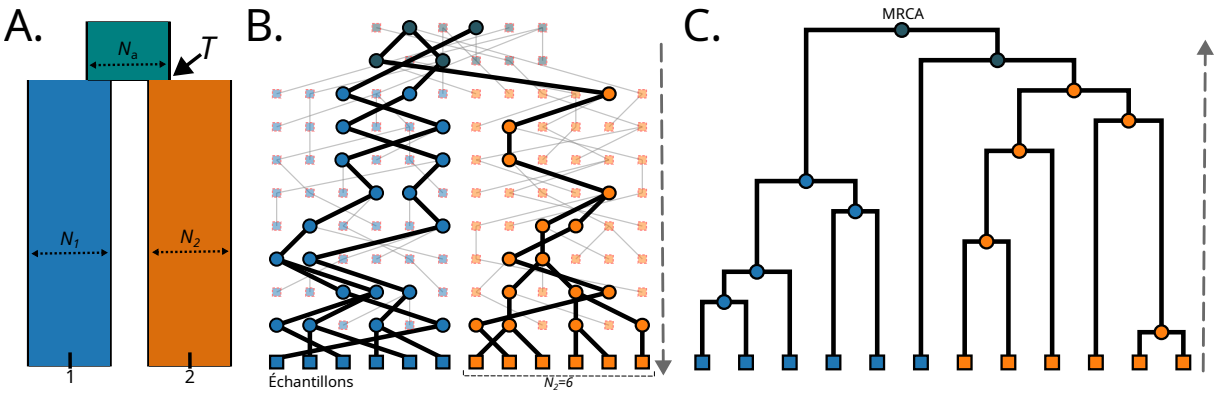

FIGURE II.4 – **Comparaison des deux paradigmes principaux de simulation de données génétiques populationnelles.**
(A) Modèle démographique de divergence en isolement de deux populations : une population ancestrale de taille efficace $N_a = 6$ se scinde, au temps T exprimé en générations, en deux populations filles, bleue et orange, de tailles efficaces respectives N_1 et $N_2 = 6$. Chaque population évolue ensuite indépendamment sans migration.
(B) Approche prospective : la simulation fait évoluer l'ensemble des individus de la méta-population, génération par génération, du passé vers le présent (flèche grise), y compris ceux n'ayant pas de descendance actuelle (représentés en semi-transparence).
(C) Approche par coalescence : la simulation reconstruit uniquement la généalogie des individus échantillonnés en remontant du présent vers le passé jusqu'au *MRCA*. Chaque nœud de l'arbre (rond) représente un événement de coalescence.

La première étape de la simulation consiste à générer des séquences génétiques populationnelles, i.e. présentant un gradient d'apparentement, selon un modèle démo-génétique. Parmi les différentes approches possibles, nous avons privilégié l'utilisation d'une méthode basée sur **la théorie de la coalescence** formalisée par **John Kingman** au début des années 1980 [17]. Cette approche, dite *backward-in-time*, modélise l'histoire généalogique d'un échantillon de manière rétrospective (fig. II.4, C) : sous l'hypothèse de *panmixie*, les lignées échantillonnées convergent progressivement par fusions successives vers leur **Ancêtre Commun le Plus Récent** (*Most Recent Common Ancestor*, ou **MRCA**). Ce

cadre fournit une description probabiliste efficace de la diversité génétique sans avoir à simuler toute la dynamique de la population. D'un point de vue computationnel, il présente deux avantages majeurs : (1) seules les lignées des individus effectivement observés sont considérées, (2) les probabilités conditionnelles de coalescence des lignées sont connues de manière analytique. Ces caractéristiques réduisent considérablement le temps de calcul des simulations. À l'inverse, le modèle *forward-in-time* (prospectif, fig. II.4, B) simule explicitement l'évolution d'une population entière du passé vers le présent et doit donc prendre en compte l'ensemble des individus et des lignées, y compris celles qui disparaissent ou ne sont pas échantillonnées, ce qui est extrêmement demandeur pour des populations de plusieurs centaines de milliers d'individus comme observé chez les arbres forestiers.

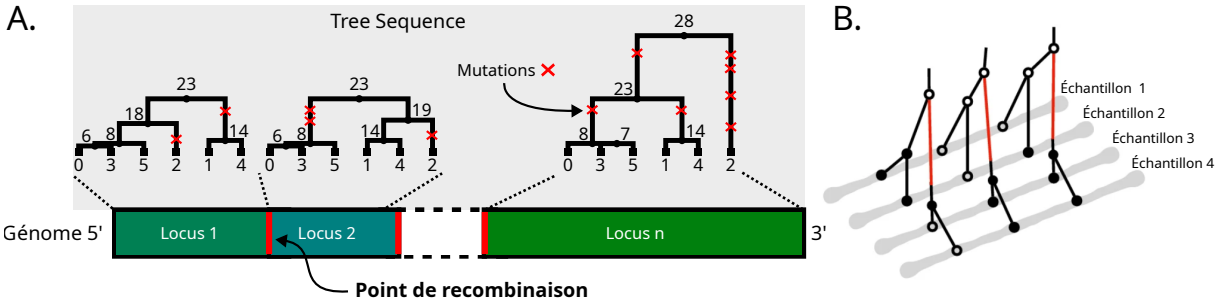

FIGURE II.5 – **Représentation de la recombinaison par la *tree-sequence*.** La *tree-sequence* est une structure de données qui encode efficacement l'ensemble des généalogies locales le long d'un génome recombiné. **(A)** La recombinaison, modélisée par le simulateur, segmente le génome en une collection de **topologies locales** (une par bloc de recombinaison). Les arbres adjacents (Locus 1 et 2), séparés par un unique événement de recombinaison (rouge), présentent des topologies très similaires. Cette forte autocorrélation reflète le **déséquilibre de liaison (LD)**. À l'inverse, le **Locus** n, physiquement distant, présente une topologie décorrélée, tendant vers l'équilibre de liaison avec les deux précédents locus. Les mutations (croix), marqueurs de l'histoire évolutive, sont superposées aux arbres proportionnellement à la longueur des branches suivant une loi de Poisson. **(B)** Visualisation des lignées ancestrales le long du génome pour 4 échantillons (d'après Deng *et al.* [18]). Les lignes rouges matérialisent les événements de recombinaison qui modifient l'ascendance locale, faisant basculer un individu d'une branche généalogique à une autre le long de la séquence. [19]

Nous avons sélectionné l'outil `msprime` [20] pour cette tache. Il s'agit d'un simulateur de coalescence de référence, reconnu pour ses performances, dont l'innovation majeure est la **séquence d'arbres** (*tree-sequence*) [19]. Cette structure stocke de manière compacte l'ensemble des arbres généalogiques le long d'un génome. Elle est indispensable pour modéliser la **recombinaison** à l'échelle génomique : un phénomène biologique clé qui fait que des positions adjacentes peuvent avoir des **histoires généalogiques distinctes**. Par conséquent, il n'existe pas une phylogénie globale unique pour le génome, mais une **mosaïque de topologies locales** (i.e. généalogies) dépendant du paysage de recombinaison génomique populationnel. Cette complexité est plus coûteuse qu'un modèle sans recombinaison reposant sur une phylogénie unique, mais elle se prête naturellement à la parallélisation des locus (voir glossaire), ce qui en limite l'impact computationnel.

B-MSpangepop) Manipulation des arbres et des graphes

Pour réaliser la conversion de la *tree-sequence* en un graphe de variation, `MSpangepop` doit manipuler deux structures de données fondamentales dont les propriétés sont très spécifiques : **l'arbre de coalescence et le graphe bi-orienté**.

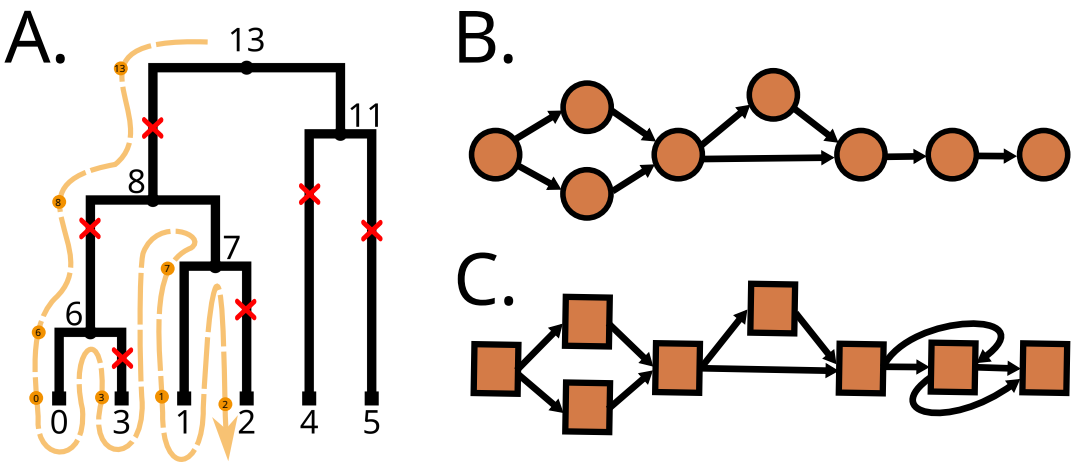

FIGURE II.6 – **Structures de données fondamentales pour la conversion des arbres de coalescence en PVG. (A)** Arbre de coalescence (issu de `msprime`). La racine (13) représente le MRCA et les feuilles (0-5) les échantillons actuels. Le chemin orange illustre le **parcours préfixe** qui visite les nœuds en respectant la chronologie des lignées. **(B)** Graphe Orienté Acyclique (DAG). Représentation classique où les arêtes ont un sens unique. Cette structure permet de modéliser certains types de variations dans des bulles (voir glossaire) mais peine à modéliser les inversions ou les duplications en tandem sans artificiellement dupliquer les nœuds. **(C)** Graphe bi-orienté. Contrairement au DAG, les arêtes connectent des extrémités spécifiques des nœuds (représentés ici par les côtés des carrés). Cette flexibilité permet de créer des cycles et de traverser un nœud dans les deux sens. On peut représenter tous les types de variations dans des snarls (voir glossaire) avec cette structure de données. (B-C adaptés de Paten *et al.* 2018 [21])

L'**arbre de coalescence** (ou généalogie, fig. II.6) issu de `msprime` est un arbre **ultramétrique** (toutes les feuilles sont équidistantes de la racine) et orienté temporellement : les feuilles représentent le « présent » ($t = 0$) et la racine est le MRCA [20]. Les longueurs de branches sont proportionnelles au temps (en générations). Les nœuds internes représentent les événements de coalescence, où deux lignées ancestrales fusionnent. Enfin, les mutations sont superposées *a posteriori* sur cet arbre, placées le long des branches selon un taux de mutation suivant une loi de Poisson. Pour exploiter cette structure temporelle, nous utilisons une **traversée d'arbre préfixe** [17] (ou parcours en profondeur). En parcourant l'arbre de la racine vers les feuilles, nous pouvons lire tous les événements (mutations) affectant une lignée dans l'ordre chronologique de leur apparition (fig. II.6 A). Cette traversée ordonnée sert de base à la construction itérative du graphe de variations.

Le graphe de variations, la structure cible, possède également des propriétés particulières. Les nœuds représentent des séquences d'ADN, qui sont intrinsèquement directionnelles (orientées 5′ vers 3′). Une lecture inverse de cette séquence (3′ vers 5′) correspond à une **inversion génomique**, que nous devons impérativement modéliser. Par conséquent, le graphe possède deux niveaux d'orientation. D'une part, il est globalement orienté d'un bout à l'autre du chromosome, formant un Graphe Orienté Acyclique (*Directed Acyclic Graph*, ou DAG). D'autre part, chaque nœud possède une orientation interne, permettant de le traverser dans le sens $5' \to 3'$ ou $3' \to 5'$. Cette structure de données est appelée un **graphe bi-orienté** (fig. II.6 C). Le format GFA, est spécifiquement conçu pour encoder ce type de graphe. Cette double orientation, bien que biologiquement nécessaire, ajoute une complexité algorithmique significative, notamment pour la représentation correcte des variants emboîtés [21].

C-MSpangepop) Paramétrage des simulations

L'ensemble des simulations repose sur une séquence ancestrale commune, fournie au format FASTA, qui constitue le patron initial de la simulation. Cette séquence représente l'état du génome avant toute divergence entre les échantillons. À partir de cette séquence de référence, `MSpangepop` simule l'évolution des génomes dans un cadre démo-génétique décrivant à la fois les caractéristiques du génome et l'histoire évolutive des populations considérées (fig. II.4 A). Ce cadre, encodé selon le format standardisé `demes`, garantit que les généalogies coalescent jusqu'au MRCA. Le génome y est caractérisé par son taux de recombinaison (r) et son taux de mutation (μ), tandis que les populations sont définies par leurs tailles efficaces (N_e) ainsi que les événements démographiques, tels que les scissions de populations, les variations de taille ou les flux migratoires, qui vont les impacter. Les séquences ainsi obtenues peuvent ensuite être façonnées grâce à des distributions de variants permettant de modéliser différents niveaux de complexité structurale. On peut, par exemple, générer des jeux de données simples n'incluant qu'un nombre restreint d'inversions à des fins de *benchmarking*. À l'inverse, on peut également simuler des pangénomes réalistes en reproduisant des distributions de tailles et de fréquences observées dans la littérature pour chaque type de SV. Les variants considérés dans nos simulations sont listés ci-dessous :

- **Insertion (INS)** : ajout de nucléotides dans la séquence d'ADN ;
- **Délétion (DEL)** : perte de nucléotides dans la séquence d'ADN ;
- **Inversion (INV)** : remplacement d'un segment d'ADN par son complément inverse, modifiant le sens de lecture (ex. : TTAG → CTAA) ;
- **Duplication en tandem (TDUP)** : répétition d'un segment d'ADN immédiatement adjacente à la séquence originale ;
- **Polymorphisme nucléotidique simple (SNP)** : substitution d'une seule base.

D-MSpangepop) Ressources utilisées pour le benchmarking

Dans le cadre des tests de développement et de validation, nous avons sélectionné des jeux de données présentant un coût computationnel limité tout en conservant un niveau de réalisme biologique satisfaisant. La séquence ancestrale utilisée correspond au génome de la bactérie *Staphylococcus aureus* [22], d'une taille d'environ 5 Mbp. Bien que de petite taille, ce génome permet de reproduire des scénarios complexes tout en maintenant des temps de calcul raisonnables. Afin de générer un paysage de variants plausibles, nous avons appliqué les distributions de tailles et de fréquences observées chez l'humain par Sudmant *et al.* (2015) [23]. Les modèles démo-génétiques utilisés ont été définis de manière *ad hoc* au cas par cas et sont décrits en chapitre III.

III | Résultats et Discussions

1. Maintien et amélioration d'asm4pg

A-Asm4pg) Simplification de l'utilisation et refonte du code

Mes objectifs étaient de simplifier l'utilisation d'`asm4pg` [12] et de garantir une chaîne de traitement robuste, y compris dans des cas d'utilisation complexes. Dans cette optique, mon premier axe de travail a porté sur la **facilité d'exécution**. Initialement, le *workflow* comportait deux étapes distinctes à exécuter : un pré-traitement des données (décompression, conversion) suivi de l'assemblage proprement dit. Ces deux étapes ont été fusionnées par souci de simplification. J'ai également développé des scripts `Bash` capables de déterminer automatiquement l'algorithme de décompression à appliquer et les conversions à réaliser pour obtenir les fichiers FASTA requis par les outils en aval. Cela permet de diminuer le nombre d'interactions utilisateur.

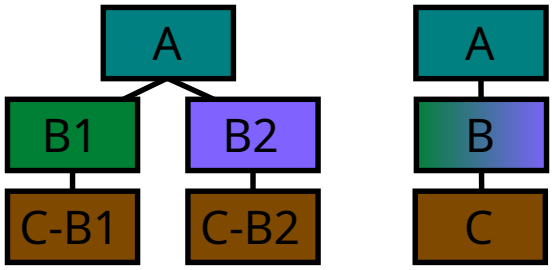

FIGURE III.1 – **Stratégie de refactorisation pour réduire la duplication de code.** (Gauche) Dans le *workflow* initial, la règle B existe en deux versions pour gérer deux modes d'exécution. La règle C, bien qu'identique, doit être dupliquée pour satisfaire les dépendances de Snakemake. (Droite) Dans le nouveau *workflow*, un script d'exécution conditionnels gère dynamiquement la règle B, ce qui affranchit le *workflow* de la duplication de code pour les règles B et C.

Mon second axe de travail visait à **garantir la robustesse** du *workflow* et à faciliter les mises à jour. Pour ce faire, j'ai réalisé une refactorisation du code. Environ la moitié du *workflow* était dupliquée pour gérer différents modes d'exécution (par exemple, des paramètres de filtre stricts ou relâchés). J'ai remplacé cette duplication par des scripts d'exécution conditionnels qui gèrent tous les cas de figure de façon unifiée. Associé à des profils de configuration pour chaque HPCC, le *workflow* complet (gestion de l'environnement, pré-traitement, assemblage) s'exécute désormais via une commande unique : `sbatch asm4pg run`. La stratégie de refactorisation est illustrée par la Figure III.1.

Élément	Version initiale	Version mise à jour	Évolution
Lignes de code	1726	1105	-35%
Règles Snakemake	48	20	-58%

TABLE III.1 – **Comparaison du code entre les versions d'Asm4pg.**

La table 1 quantifie l'impact de cette réécriture : le nombre de règles du *workflow* a été réduit de 58%. Cette optimisation structurelle n'a pas été réalisée au détriment des capacités de l'outil, mais a au contraire permis de préparer l'augmentation de ses fonctionnalités, présentées dans la section suivante. Cette refactorisation était d'autant plus essentielle que le *workflow* gère désormais 4 modes (un mode par défaut et trois alternatifs) et 6 scripts conditionnels, générant près de 160 combinaisons d'exécution distinctes. Conserver l'approche initiale par duplication de code aurait rendu la maintenance du *workflow* intenable. La nouvelle architecture a été validée par des tests ciblés où j'ai couvert une dizaine des cas d'exécution classiques, ce qui a permis de résoudre les instabilités liées aux combinaisons de paramètres complexes définies par l'utilisateur.

B-Asm4pg) Mises à jour et nouvelles fonctionnalités

Afin de moderniser `asm4pg`, j'ai procédé aux mises à jour des programmes existants et ai intégré de nouveaux outils améliorant notablement le *scaffolding* et le QC des assemblages (fig. III.2). J'ai également totalement réécrit la documentation afin d'y intégrer ces changements [12]. Parmi les mises à jour essentielles, le passage de l'assembleur `Hifiasm` [5] de la v`0.16.1` (2021) à la v`0.24.0-r703` (2025) est central. La nouvelle version intègre des améliorations algorithmiques majeures, notamment pour la gestion native des données Hi-C et ONT. D'autre part, le *workflow* a été enrichi de 8 nouveaux outils (pré-traitement, assemblage et QC) et du nouveau mode d'exécution **Hi-C** (fig. III.2), cette technologie fournissant une information de localité améliore drastiquement le *scaffolding*. Pour ce mode, j'ai intégré l'enchaînement suivant : `Hifiasm` assemble les lectures longues en contigs, `fastp` filtre les lectures Hi-C qui sont ensuite alignées sur les contigs par `bwa mem`. Finalement, `YaHS` (`0.4.3`) utilise cet alignement pour ordonner les contigs en pseudo-chromosomes. Ce mode, produisant des génomes d'excellente qualité, a été utilisé pour assembler la référence du projet Pangenoak.

C-Asm4pg) Test du workflow

Pour valider l'outil en conditions réelles, nous avons sélectionné le génome du Chêne pédonculé (*Quercus robur*) comme cas d'étude pour tester le mode Hi-C. Cet organisme combine deux défis majeurs pour l'assemblage : une forte hétérozygotie et un génome riche en éléments transposables (~50%). Les 4,5 millions de lectures HiFi représentant environ 20 Gb de données, soit une couverture de $\approx 80\times$, ont été assemblées par `Hifiasm` en 9 heures (avec 20 cœurs et 250 Gb de mémoire). La taille de l'assemblage obtenu (0,830 Gb) est cohérente avec la littérature et les prédictions de `Genometools`. Nous avons également assemblé le génome en mode par défaut afin d'évaluer l'apport du mode Hi-C. La contiguïté s'améliore avec le mode Hi-C puisque le $L90$ (nombre minimal de séquences nécessaires pour représenter 90% de la taille du génome) passe de 18 contigs à 12 *scaffolds* pour l'Haplotype 1, et de 21 à 11 pour l'Haplotype 2. En parallèle, le nombre total de séquences a été réduit : l'Haplotype 2 passant de 274 contigs à 112 *scaffolds* et l'Haplotype 1 passant de 1 086 contigs à 877 *scaffolds* (tab III.2).

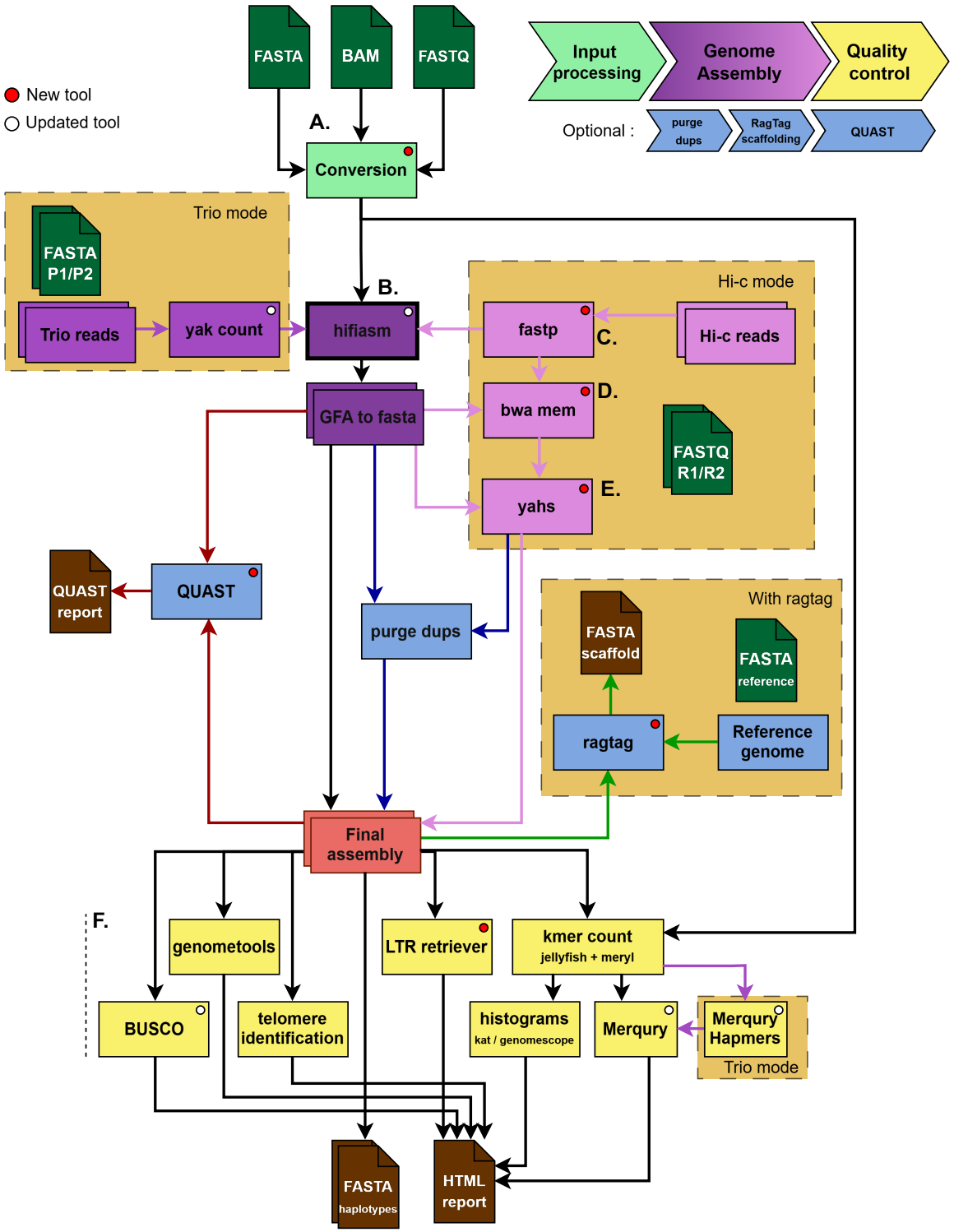

FIGURE III.2 – **Schéma fonctionnel du *workflow* `Asm4pg` (v2.0).** La figure illustre l'architecture mise à jour, organisée en trois modules : traitement des entrées (vert), assemblage (violet) et contrôle qualité (jaune). Les modes d'exécution alternatifs sont encadrés en orange. Les options sont en bleu et les sorties du workflow en marron. Pour le nouveau mode Hi-C, les lectures longues sont converties au format FASTA (A), puis assemblées en contigs par `hifiasm`. Parallèlement, `fastp` (C) prétraite les lectures Hi-C. Les lectures Hi-C traitées sont ensuite alignées par `bwa mem` sur les contigs (D). Finalement, `YaHS` (E) utilise cet alignement pour réaliser le *scaffolding*, produisant l'assemblage final qui partira en contrôle qualité (F).

	Haplotype 1		Haplotype 2	
Métrique	*Hifiasm*	+ Hi-C	*Hifiasm*	+ Hi-C
Scaffolds	1 086	877	274	112
Taille totale (Gb)	0.849	0.849	0.804	0.810
N50 (Mb)	55	62	35	68
L90	18	12	21	11
Score LAI	22.81	22.78	24.72	23.58

TABLE III.2 – **Impact du *scaffolding* Hi-C sur les métriques d'assemblage de *Quercus robur*.** Le tableau présente les métriques de contiguïté et de qualité. Par rapport à l'assemblage initial (*Hifiasm*), l'étape de *scaffolding* (+ Hi-C) se traduit par une augmentation de la valeur de *N50* (longueur telle que 50% de l'assemblage est contenu dans des séquences de cette taille ou plus) et une diminution du $L90$, tandis que les scores de complétude des répétitions (LAI) et la taille totale demeurent stables.

D-Asm4pg) Analyse des performances et positionnement

La modernisation d'`asm4pg` a transformé le *workflow* en un outil de production robuste, permettant de générer de manière automatisée des assemblages de haute qualité. Le mode par défaut permet déjà d'obtenir, chez le chêne pédonculé, des assemblages d'excellente qualité, avec un $L90$ inférieur à 24 et un $N50$ compris entre 35 et 55 Mb (tab III.2). Ce résultat témoigne d'une excellente contiguïté : il signifie que 90% du génome est contenu dans des contigs correspondant des chromosomes entiers. Le nombre élevé de *scaffolds* s'explique lui par la présence de nombreuses lectures orphelines, par exemple issues de contaminations, non rattachables aux autres contigs et attribuées par convention à l'haplotype 1. Cela explique la forte différence observée entre haplotypes et rend le nombre de *scaffolds* peu pertinent pour évaluer la qualité des assemblages à notre échelle. En mode Hi-C, la contiguïté est encore meilleure avec un $L90$ proche du nombre de chromosomes ($n = 12$) et notamment une valeur de $N50$ doublée pour l'haplotype 2 (68 versus 35). De plus, la taille cumulée des douze plus grands *scaffolds* avoisine les 800 Mb, une valeur très proche de la taille totale du génome estimée par `GenomeScope`. Ce type d'assemblage à l'échelle chromosomique fournit un cadre idéal aux approches de pangénomique, la contiguïté observée permettant de limiter la complexité des graphes et d'assurer la précision des alignements multiples.

Toutefois, il convient de situer `asm4pg` dans le paysage actuel des assembleurs disponibles. Des initiatives majeures, telles que le *Vertebrate Genomes Project* (VGP), ont développé des *workflows* reposant sur des solutions similaires [24]. La comparaison des outils met en évidence une convergence méthodologique entre `asm4pg` et le *workflow* du VGP. En effet, 9 des 17 logiciels utilisés pour l'assemblage sont communs aux deux solutions et les 4 outils spécifiques à chaque *workflow* reposent sur des principes fonctionnels similaires. Toutefois, la solution du VGP se limite à l'étape d'assemblage, le contrôle qualité étant assuré par le *workflow* du DTOL [15], une solution hautement optimisée mais non publiée. `asm4pg` se distingue également par sa capacité à traiter de manière unifiée des données hétérogènes (HiFi, Hi-C, ONT), réduisant les besoins de traitements *ad hoc*. Contrairement aux standards souvent orientés vertébrés, il a également été conçu pour répondre aux défis des génomes végétaux via l'intégration d'outils comme `LTR_Finder`. Enfin et surtout, `asm4pg` respecte strictement les bonnes pratiques de programmation décrites plus haut et offre une portabilité élevée grâce à la conteneurisation complète des outils et à la modularité du *workflow Snakemake*. Cette architecture permet un déploiement fiable sur tout type de cluster de calcul, y compris en l'absence d'une instance *Galaxy*, plateforme web dont la dépendance constitue un prérequis du *workflow* VGP.

2. Développement de MSpangepop

L'objectif principal en développant `MSpangepop`[13] était de créer un simulateur de pangénomes satisfaisant quatre propriétés fondamentales :

1. La prise en compte de **scénarios démographiques** permettant de modéliser de manière réaliste les relations d'apparentement entre individus.
2. La représentation fidèle du **paysage de recombinaison** permettant de modéliser l'hétérogénéité généalogique du génome.
3. La possibilité de modéliser des **variants complexes** emboîtés ou chevauchants.
4. La génération native de graphes de pangénome (4.1) au format **GFA** (4.2).

Le projet a connu plusieurs itérations majeures, avec des changements structurels profonds et des réorientations méthodologiques successives. Après stabilisation du cahier des charges et identification claire des limitations techniques, j'ai développé une architecture de *workflow* modulaire servant de base solide pour les extensions futures. Cette section détaille la structure et l'implantation de cet outil.

A-MSpangepop) Architecture du simulateur de pangénomes

Propriétés 1 et 2 : modèles démo-génétiques et paysage de recombinaison

Afin de satisfaire aux propriétés 1 et 2, nous utilisons le programme `msprime` qui simule par coalescence des séquences de généalogies le long du génome conformément à un modèle démo-génétique donné (fig. II.4) [25]. La rapidité d'exécution et le réalisme des simulations reposent sur un ensemble de paramètres clés, que l'on peut regrouper en deux catégories principales : (i) les **paramètres démographiques** et (ii) les **paramètres d'échantillonnage et génomiques**.

Les paramètres démographiques spécifient l'histoire évolutive des populations considérées. Ils incluent notamment :

— le nombre de populations, noté i ;
— la taille efficace de chaque population, $N_{e,i}$, qui détermine l'intensité de la dérive génétique ;
— des événements démographiques comme la migration, la scission de populations ou les changements de taille des populations dans le temps (e.g. goulot d'étranglement).

Les paramètres d'échantillonnage et génomiques définissent quant à eux les caractéristiques des données simulées et le cadre moléculaire sous-jacent. Ils comprennent :

— le nombre d'individus échantillonnés, et par conséquent le nombre de lignées simulées, noté n ;
— la longueur du génome simulé, L ;
— le taux de mutation par paire de bases et par génération, μ ;
— le taux de recombinaison par paire de bases et par génération, r.

Ce dernier terme intervient notamment dans le taux de recombinaison populationnel $\rho = 4N_eLr$ directement responsable du nombre de recombinaisons, et donc de généalogies, le long des génomes. Historiquement, ce nombre était le facteur limitant des simulations de coalescence puisque l'algorithme classique de Hudson, implanté dans `ms` [26], présentait une complexité **quadratique** par rapport au taux de recombinaison ($\mathcal{O}(\rho^2)$). L'utilisation de la structure de données *tree-sequence* (fig. II.5) par `msprime`, permettant le partage des structures invariantes entre généalogies successives, réduit la complexité algorithmique pour la rendre quasi linéaire par rapport à ρ et à n [20]. Toutefois, la simulation de génomes entiers implique mécaniquement un grand nombre d'événements de recombinaison, proportionnellement à L. Afin de limiter l'impact computationnel de cette contrainte, notre architecture exploite l'indépendance biologique des chromosomes en considérant leur évolution comme autonome. Les simulations sont ainsi **parallélisées** chromosome par chromosome, chacun ayant sa propre *tree-sequence*, ce qui permet de modéliser fidèlement la recombinaison tout en optimisant le temps d'exécution global.

Le *workflow* `Snakemake` orchestre l'ensemble du processus de simulation populationnelle. Il récupère le génome servant de séquence ancestrale au format **FASTA**, indexe la taille de chaque chromosome (fournissant ainsi L), puis lance une simulation `msprime` distincte pour chacun. Le modèle démographique et les paramètres d'échantillonnage additionnels sont définis par l'utilisateur dans un fichier au format **JSON**, dynamiquement traduit en paramètres `msprime`. Ce format est très simple à traiter en `Python`, permettant la sérialisation de structures complexes. Pour faciliter les analyses statistiques en aval, j'ai implanté un système de **réplicats** permettant de lancer de multiples simulations avec des paramètres fixes ou tirés aléatoirement dans des intervalles définis. Enfin, l'ensemble du processus peut être initialisé par une **graine aléatoire** (*seed*) pour éliminer les composantes stochastiques et garantir la reproductibilité.

Propriété 3 : modélisation de variants complexes

Une fois les arbres généalogiques obtenus par simulation pour chaque chromosome, une traversée préfixe de chaque arbre permet d'obtenir la liste des mutations ordonnées chronologiquement (fig. II.6.A). Cette approche simple et rapide, présentant une complexité en $\mathcal{O}(N)$ où N est le nombre de nœuds, garantit qu'une mutation survenue à un temps plus ancien soit toujours traitée avant une mutation plus récente. L'algorithme que j'ai implanté pour cette traversée est récursif ; les dimensions des arbres simulés ne posant jamais de problème de dépassement de la pile d'exécution (*stack overflow*). La traversée retourne pour chaque arbre une liste de mutations, leurs positions sur l'arbre et sur le chromosome. Cette liste est enregistrée dans un fichier JSON intermédiaire appelé « Traversée ». Si cette structure de données, organisée en liste chronologique, entraîne une perte des relations **explicites** de parenté au sein des arbres, elle facilite par ailleurs les traitements ultérieurs des lignées, notamment car la liste des descendants affectés par chaque mutation a été sauvegardée pendant la traversée permettant d'identifier instantanément l'ensemble des individus qui sont affectés par un variant. Chaque traversée est ensuite analysée de manière parallèle, toutes les mutations présentes sur un locus ℓ étant traitées par un *thread* avant d'être unies au niveau chromosomique.

Pour convertir les mutations en variants génétiques concrets, nous leur attribuons un type de variation, une position et une taille en itérant sur la liste et en tirant aléatoirement tous les paramètres à partir de distributions définies *a priori* en suivant l'Algorithme 1.

Algorithme 1 Conversion des mutations en variants et propagation des coordonnées

pour tout locus ℓ **faire**
  $\mathcal{M} \leftarrow$ Liste des mutations sur ℓ triée par temps (ancien $\rightarrow$ récent)
  pour toute mutation $m \in \mathcal{M}$ **faire**
    $[i_m, j_m] \leftarrow m$.intervalle {Intervalle génomique du locus dans son état courant}
    $\tau \leftarrow$ TirerType(P_{variants})
    $p \leftarrow$ TirerPosition(i_m, j_m) {Position relative dans l'intervalle courant}
    $k \leftarrow$ TirerTaille(τ, distribution$_\tau$)
    {Propagation des changements de coordonnées}
    pour toute mutation m_{next} après m dans $\mathcal{M}$ **faire**
      Mettre à jour m_{next}.intervalle en fonction du type τ et de la taille k
    fin pour
  fin pour
fin pour

L'aspect crucial de cet algorithme est la **mise à jour dynamique des intervalles de définition** des locus. Lorsqu'un variant de taille k est introduit à la position p, l'intervalle $[i, j]$ du locus change pour toutes les mutations qui surviendront ultérieurement (c'est-à-dire les mutations plus récentes dans la liste $\mathcal{M}$). Cette propagation garantit la cohérence topologique des variants emboîtés. En effet, si une partie du matériel génétique est supprimée au temps ancien (i.e. délétion), aucune mutation ne pourra survenir sur les bases perdues au temps récent. Inversement, si de nouvelles bases apparaissent (i.e. insertion), elles constituent une nouvelle cible pour les variations futures, permettant nativement de générer des **variants emboîtés**. Cependant, elle induit un coût algorithmique important. Puisque chaque modification de coordonnées doit être répercutée sur l'ensemble des mutations suivantes dans la liste, le traitement d'une lignée comportant m_l mutations entraîne une complexité locale en $\mathcal{O}(m_l^2)$. À l'échelle du génome, la complexité globale reste donc bornée par $\mathcal{O}(m^2)$ dans le pire des cas.

Propriété 4.1 : construction du graphe pangénome

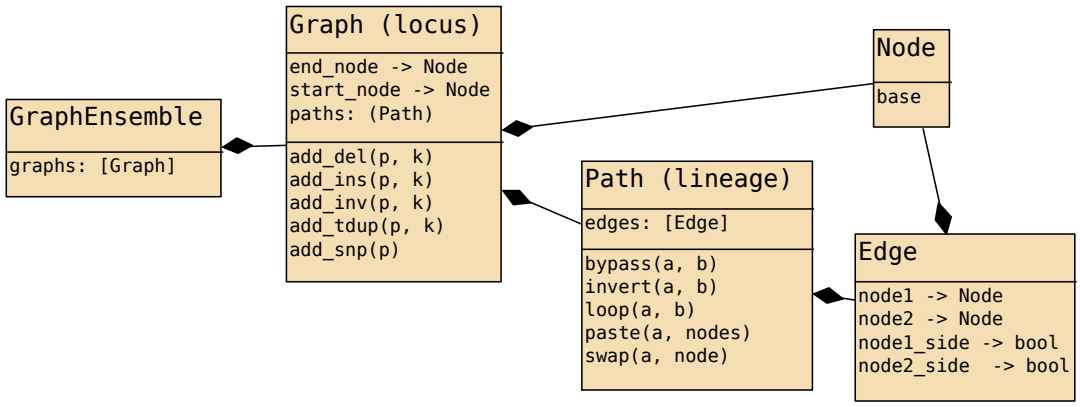

FIGURE III.3 – **Diagramme UML simplifié de MSpangepop :** illustre la hiérarchie des objets : les flèches symbolisent les relations de composition (relation contenant-contenu). Le compartiment inférieur de chaque bloc détaille les méthodes principales et leurs arguments.

Une fois calculée, la « Traversée », enrichie des informations de variants, sert de liste d'instructions pour l'algorithme de construction de graphe que j'ai implanté *de novo*. Afin de simplifier le développement incrémental et favoriser une forte modularité, cette implantation a été réalisée selon une approche orientée objet sous forme d'une bibliothèque Python nommée MSpangepop (fig. III.3). Celle-ci est structurée autour de plusieurs classes principales :

- les **Nodes (Nœuds)** : l'élément atomique de notre structure. Dans notre approche, **chaque nœud contient une seule base** d'ADN. Cette structure est dite *chopped* ou hachée ;
- les **Edges** : arêtes du graphe, qui sont **bi-orientées** (fig. II.6). Elles ne relient pas directement deux nœuds, mais leurs *extrémités*, notées 5' (ou `False`) et 3' (ou `True`). Une arête orientée dans le sens de la lecture reliant les nœuds A et B correspond ainsi à une connexion entre l'extrémité 3' de A et l'extrémité 5' de B ; elle est représentée par `Edge(A*, True, B*, False)`. Cette représentation est analogue à celle utilisée par `BCALM 2` [2] pour les graphes de De Bruijn bi-orientés ;
- les **Paths (Chemins)** : ils représentent les haplotypes des individus simulés. Un `Path` est une liste **ordonnée** d'arêtes (`Edges`). Les différents chemins partagent les mêmes objets `Nodes` lorsque les haplotypes respectifs partagent les mêmes allèles ce qui permet d'économiser la mémoire ;
- les **Graphs (Graphe)** : Représente le sous-graphe pour un locus donné. Il contient une référence aux chemins (`Paths`) de tous les individus ;
- le **GraphEnsemble** : Représente un chromosome complet. C'est une collection ordonnée de `Graph` (un par locus), qui gère la connectivité entre les sous-graphes successifs.

Une étape d'initialisation crée et utilise ces objets afin de mettre en place la structure fondamentale du graphe de variation, comme détaillé dans l'Algorithme 2 : pour chaque locus, cette procédure construit un sous-graphe dans lequel tous les individus partagent initialement un chemin unique correspondant à la séquence ancestrale. Les nœuds sont alors reliés de manière strictement linéaire, formant une structure en « collier de perles » (fig. III.4.B). La charge computationnelle du processus est répartie sur de multiples *threads*, chacun gérant un lot de sous-graphes.

L'intégration des variants se fait ensuite locus par locus. Pour ce faire, les méthodes de la bibliothèque `MSpangepop` opèrent directement sur les sous-graphes en suivant l'ordre chronologique défini par la traversée (alg. 3). Dans cette architecture, la classe `Graph` agit comme un orchestrateur. Elle traduit les positions génomiques du variant en identifiants de nœuds du graphe (nœud de départ n_{start} et nœud de fin n_{end}), identifie les lignées porteuses de la mutation, puis délègue l'application effective de la modification aux objets `Path` concernés. Chaque lignée maintient ainsi son propre chemin à travers les nœuds partagés, permettant des modifications indépendantes tout en conservant une structure globale cohérente. Les modifications topologiques réelles sont implantées au sein de la classe `Path` qui dispose d'un vaste ensemble de méthodes capables d'altérer la liste des arêtes pour matérialiser les variations génomiques. Ce module central gère l'ensemble des cas topologiques, notamment les variants complexes ou imbriqués (e.g. duplication au sein d'une inversion), tout en garantissant la connectivité du chemin. Cette complexité algorithmique a nécessité de prendre en compte de nombreuses configurations afin de s'assurer de la robustesse du processus (fig. III.4 pour des exemples simples). Néanmoins,

Algorithme 2 Initialisation du graphe de variation global

```
pour tout locus ℓ avec intervalle [i, j] faire
    Créer un sous-graphe G vide
    nœud_précédent ← null
    pour position p de i à j faire
        nœud ← Nœud(base[p]) {Création du nœud pour la base}
        si nœud_précédent ≠ null alors
            Créer arête (nœud_précédent → nœud) {Liaison 3' vers 5'}
        fin si
        nœud_précédent ← nœud
    fin pour
    chemin_ancestral ← séquence d'arêtes créées
    pour tout individu k faire
        G.chemins[k] ← chemin_ancestral {Partage du chemin initial}
    fin pour
fin pour
```

la diversité des configurations étant difficile à anticiper exhaustivement, un algorithme de secours intervient en cas d'imprévu : si un variant compromet la connectivité du sous-graphe, la modification est annulée et consignée dans un journal. Ce mécanisme garantit un fonctionnement fiable y compris lors de la génération de millions de sous-graphes. En conséquence, la complexité temporelle *par sous-graphe*, correspondant à l'initialisation du graphe (alg. 2) et à l'application des variants (alg. 3), est $\mathcal{O}(n \cdot (L_{\text{locus}} + V_{\text{locus}}))$ avec L_{locus} la longueur du locus, V_{locus} le nombre de variants et n le nombre d'individus. Le paramètre le plus critique est n : pour des valeurs modérées, la complexité reste quasi linéaire dans des scénarios biologiques où $n \approx V_{\text{locus}} \ll L_{\text{locus}}$. À l'échelle du pangénome complet, la complexité globale correspond à la somme de ce coût sur l'ensemble des locus, soulignant l'importance critique du taux de recombinaison populationnel ρ.

Algorithme 3 Application des variants sur les chemins d'un sous-graphe

```
pour tout locus ℓ avec un sous-graphe G faire
    pour tout variant v défini par (type τ, position p, longueur k) faire
        C ← identifier_chemins_affectés(descendants)
        pour tout chemin c ∈ C faire
            essayer :
            (n_a, n_b) ← convertir_position(c, p, k) {Conversion index → positions sur c}
            c.M_τ(n_a, n_b) {Application de la modification sur le chemin}
            Valider la connectivité du graphe
            si échec : Annuler l'opération et journaliser l'erreur
        fin pour
    fin pour
fin pour
```

Propriété 4.2 : Sauvegarde au format GFA

Une fois la topologie locale résolue pour chaque locus, l'enjeu est d'assembler les sous-graphes en un pangénome chromosomique complet. Pour limiter l'empreinte mémoire,

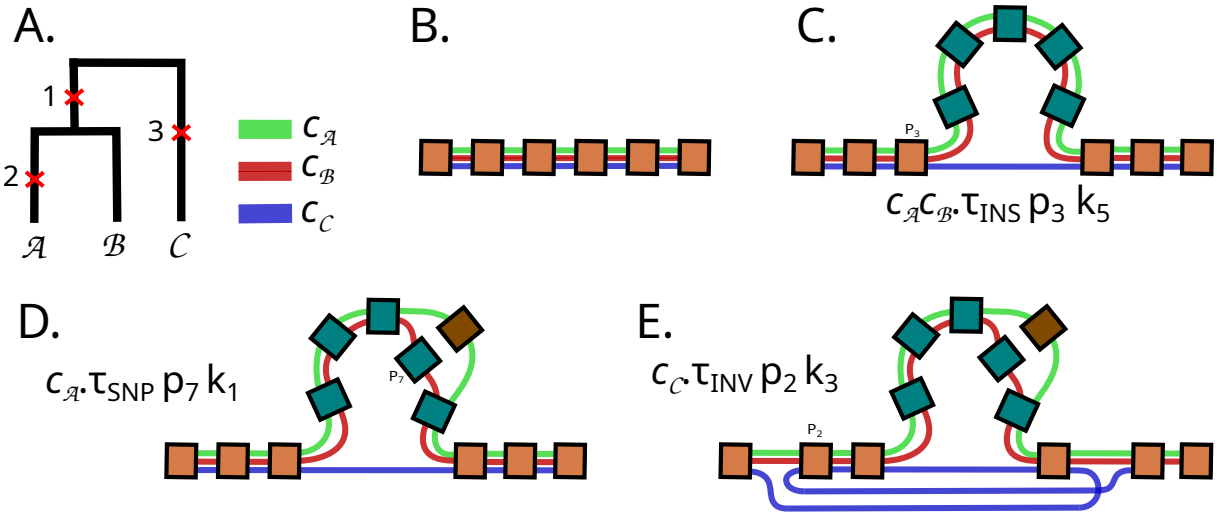

FIGURE III.4 – **Illustration de l'application séquentielle des variants sur les chemins du graphe. (A)** La « Traversée » de l'arbre généalogique dicte l'ordre de traitement des mutations 1 à 3 pour les trois haplotypes $(c_{\mathcal{A}}, c_{\mathcal{B}}, c_{\mathcal{C}})$, ancestralement identiques **(B)** et partageant donc le même chemin unique. **(C)** Une insertion ancestrale (mutation 1) de taille k_5 est ajoutée à la position p_3. Elle crée de nouveaux nœuds (turquoises) et modifie les chemins $c_{\mathcal{A}}$ et $c_{\mathcal{B}}$. **(D)** Une mutation ponctuelle (mutation 2) survient spécifiquement sur la lignée A créant un SNP avec la lignée $\mathcal{B}$. Notons que la position p_7 est **relative** au chemin $c_{\mathcal{A}}$ précédemment modifié, créant ainsi des variants emboîtés. **(E)** Une inversion (mutation 3) affecte la lignée $\mathcal{C}$. Aucun nouveau nœud n'est créé ; seules les arêtes sont altérées, créant une boucle topologique (chemin bleu) sur les nœuds existants.

chaque sous-graphe est initialement stocké dans un fichier temporaire. L'assemblage final du chromosome s'opère ensuite par une concaténation de proche en proche : le dernier nœud d'un locus est connecté au premier du suivant. Cette approche modulaire impose une contrainte stricte : les positions aux extrémités de chaque locus doivent être préservées de toute mutation afin de garantir une jonction parfaite et la connectivité globale du graphe.

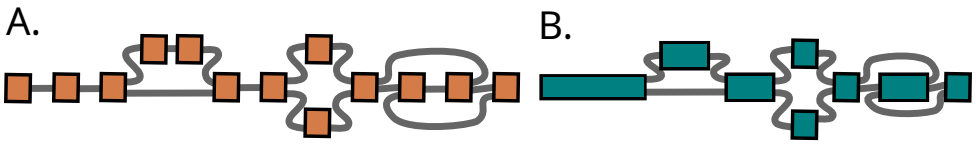

FIGURE III.5 – **La compression du graphe (ou *unchopping*).** (A) Le graphe initial en version *chopped*, chaque nœud représente une base. (B) Le graphe final *unchopped*, les séquences linéaires (unitigs) ont été fusionnées.

Le graphe brut ainsi assemblé est toujours sous forme dite *chopped*. Cette granularité extrême génère un volume de données massif, rendant la sérialisation au format GFA critique. Pour répondre à ce défi, j'ai implanté une stratégie d'écriture parallélisée : les définitions de nœuds (lignes S) et la reconstruction des chemins et arêtes (lignes P et L) sont gérées par des processus distincts. Les données sont agrégées en mémoire tampon avant d'être écrites sur le disque en une opération séquentielle unique. Cette méthode optimise les entrées/sorties (I/O) et permet d'attribuer les identifiants numériques finaux uniquement lors de l'export, allégeant considérablement la consommation mémoire durant le traitement. En parallèle, un fichier FASTA contenant les séquences des haplotypes simulés est également généré. Le GFA *chopped* généré nécessite ensuite d'être compressé en fusionnant les chaînes linéaires de nœuds (i.e. unitigs) en nœuds uniques afin de réduire drastiquement la complexité du graphe (fig. III.5). Cette étape de post-traitement, appelée

unchopping, est complexe à réaliser pour des graphes bi-orientés. Afin de simplifier le traitement, nous utilisons l'outil `VG` (Variation Graph toolkit), qui propose la commande `vg mod --unchop`. Implantée en C++, elle utilise un algorithme glouton pour identifier et fusionner les régions linéaires du graphe. Elle réduit la taille du fichier GFA d'environ 99,7% et préserve intégralement la topologie. C'est cette étape d'*unchopping* qui est la plus gourmande de notre workflow en temps de calcul, sa complexité en $O(P \times N)$ impliquant, pour un graphe de 10^{10} nœuds (N) et 5 chemins (P), un temps d'exécution de l'ordre de 2 à 4 heures (voir code source).

B-MSpangepop) Outils d'analyse des données simulées

Afin de faciliter l'analyse des simulations et du pangénome généré, le workflow intègre des scripts produisant de nombreuses figures et fichiers de journalisation, parmi lesquels deux sorties principales sont présentées ci-dessous.

La visualisation directe d'une *tree-sequence* contenant plusieurs centaines d'arbres distincts étant impraticable, un module de visualisation basé sur la méthode **STEAC** (*Species Tree Estimation using Average Coalescence*) a été développé. Cette approche consiste à calculer, pour chaque paire d'individus, l'âge moyen de coalescence à travers l'ensemble des locus, produisant ainsi une matrice de distances génétiques moyennes. Cette matrice permet ensuite de reconstruire un arbre de consensus représentatif de l'histoire évolutive globale. Cette fonctionnalité n'étant pas disponible nativement dans la bibliothèque `msprime`, l'algorithme a été implanté en Python. Ce développement permet à la fois de produire des visualisations synthétiques, comme celle présentée en Figure III.6.B, et de fournir une méthode simple et robuste pour résumer une *tree-sequence* complexe.

En complément des visualisations, `MSpangepop` génère un fichier de journalisation exhaustif (*log*) recensant l'ensemble des mutations introduites lors de la simulation. Chaque entrée précise le type de mutation, sa position génomique, le locus concerné ainsi que les lignées affectées. Ce registre constitue une référence très utile pour l'évaluation de la sensibilité et de la précision des outils de détection des variants opérant sur des graphes de pangénomes. L'objectif de ces outils est d'identifier les variations génétiques d'un individu donné par rapport à l'ensemble des chemins représentés dans le graphe.

C-MSpangepop) Preuve de concept : simulation d'un pangénome de *Staphylococcus aureus*

Afin de tester `MSpangepop` dans un contexte biologique réaliste, j'ai réalisé une simulation basée sur le génome de *Staphylococcus aureus* (~3 Mpb). Le scénario démographique retenu modélise une divergence simple entre deux populations 1 et 2 pouvant correspondre à un repiquage de bactéries (fig. III.6.A). Les paramètres utilisés sont les suivants :

- **taux de mutation** (μ) : 1×10^{-8} ;
- **taux de recombinaison** (r) : 1×10^{-9} ;
- **tailles efficaces des populations** (N_i) : $N_1 = 5\,000$ et $N_2 = 500$ individus ;
- **temps de divergence** (T) : $T = 3\,000$ générations ;
- **échantillonnage** (n_i) : $n_1 = 18$ et $n_2 = 6$ soit un total de 24 individus haploïdes.

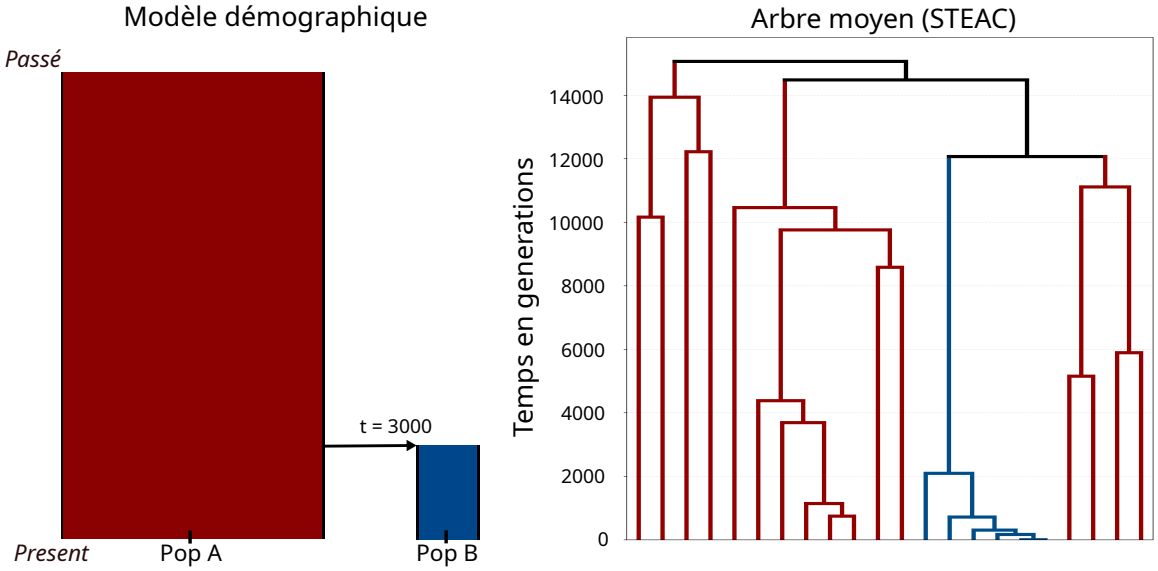

FIGURE III.6 – **Modélisation démographique et généalogie moyenne du pangénome simulé de *Staphylococcus aureus*.** (A) Scénario démographique. La population 2 (bleue) est issue d'un événement de divergence (e.g. repiquage) depuis la population 1 (rouge) au temps $T = 3\,000$ générations. (B) Arbre STEAC calculé à partir de l'ensemble des locus de la simulation. Les lignées des populations 1 et 2 sont représentées en rouge et bleu, respectivement.

À l'issue de la simulation, nous obtenons un pangénome fragmenté en 240 arbres généalogiques (locus), portant un total de 2 603 mutations. D'un point de vue démographique, l'analyse STEAC de ces locus (fig. III.6.B) montre, comme attendu, que les individus de la population 2 (lignées bleues) coalescent beaucoup plus rapidement que ceux de la Population 1 (lignées rouges). Cela s'explique par sa plus faible taille efficace N_2 qui accélère la dérive génétique et la fixation des allèles. Parallèlement, le tri de lignées ancestrales de la population 1 est pour le moment incomplet, la faisant apparaître polyphylétique (i.e. constituée de plusieurs lignées évolutives).

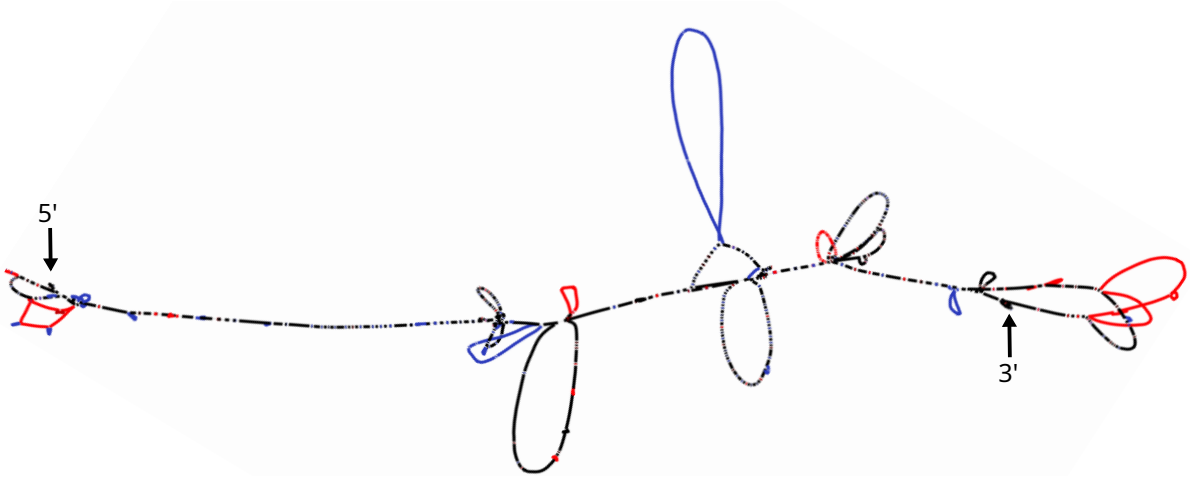

FIGURE III.7 – **Visualisation du graphe de pangénome avec Bandage.** Les nœuds sont colorés selon leur spécificité populationnelle : les blocs noirs correspondent au génome cœur partagé par les deux populations, les blocs rouges sont spécifiques à la Population 1 et les bleus à la Population 2.

Le pangénome simulé étant de petite taille, nous pouvons le visualiser entièrement grâce au logiciel `Bandage` (fig. III.7). La visualisation obtenue révèle une structure to-

pologique complexe et réaliste où on distingue clairement des régions conservées (nœuds *core*) et des bulles de variations spécifiques à chaque population. Plus notable encore, la présence de variations emboîtées et de structures denses témoignent de la capacité de l'outil à simuler des variants complexes. Cette complexité structurelle est comparable à celle observée dans des pangénomes biologiques réels.

D-MSpangepop) Benchmarking d'un outil de détection de variants

Afin de valider la pertinence de `MSpangepop` comme générateur de données de référence, j'ai réalisé le *benchmarking* de `INVPG annot` [27], un outil de détection des inversions à partir de PVG. Le fonctionnement de cet outil se décompose en trois étapes clés :

- **Déconstruction :** projection du graphe sur un chemin de référence linéaire grâce à l'outil externe `vg deconstruct` [28], produisant une représentation du pangénome sous forme d'un fichier VCF ;
- **Identification des inversions explicites :** repérage, dans le fichier VCF, des inversions explicitement décrites et correspondant à des événements directement encodés dans la topologie du graphe (*path explicit*).
- **Récupération des inversions non explicites :** alignement sur le graphe des variants restants (*Alignment-rescue*) pour détecter les inversions non explicitement encodées suite à d'éventuelles erreurs de construction du PVG (e.g. une inversion de grande taille incorrectement représentée comme une insertion).

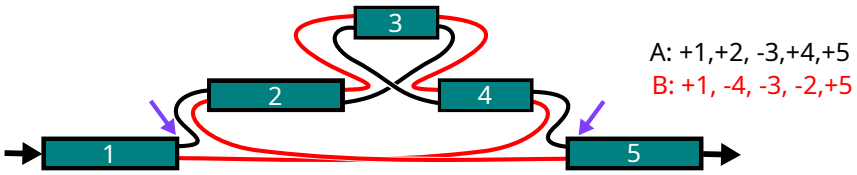

FIGURE III.8 – **Topologie d'une inversion imbriquée illustrant une bulle difficile à caractériser.** L'outil de de détection de variants identifie correctement la bulle de variation globale entre les nœuds 1 et 5 (délimitée par les flèches violettes), mais échoue à résoudre la structure interne. Sur le plan généalogique, ce cas est complexe : le chemin A (noir) porte une inversion locale du nœud 3 $(+2, -3, +4)$, tandis que le chemin B (rouge) porte une inversion majeure couvrant l'ensemble du segment 2-4 $(+1, -4, -3, -2, +5)$. En conséquence, bien que les deux haplotypes traversent le nœud 3 dans la même orientation (inversée), les nœuds adjacents 2 et 4 sont opposés l'un à l'autre, créant un variant emboîté non réductible par les approches linéaires standard.

J'ai simulé via `MSpangepop` un graphe contenant exclusivement des inversions, avec un total de 55 événements distincts répartis entre deux haplotypes. L'analyse du graphe par `INVPG annot` a permis de détecter 48 inversions avec le mode *path explicit* et 0 avec le mode *Alignment-rescue*. Cette absence de détection par la procédure de « rattrapage » (*rescue*) est attendue dans notre contexte. En effet, `MSpangepop` construit le graphe directement à partir de la généalogie et des mutations simulées, sans passer par les étapes d'alignement ou d'assemblage sujettes aux erreurs. La topologie du graphe est donc théoriquement parfaite, rendant les inversions explicitement lisibles dans les chemins (*paths*). Concernant les sept inversions non détectées par `INVPG annot`, une inspection manuelle a confirmé que toutes correspondaient à des variants complexes : des inversions **emboîtées** (une inversion au sein d'une autre, fig. III.8) ou **chevauchantes**. À l'inverse, toutes

les inversions isolées ont été parfaitement identifiées. Ce comportement s'explique par une limitation intrinsèque de l'étape de déconstruction du graphe par `vg deconstruct` (détection initiale des variants) qui ne permet pas une résolution récursive des bulles complexes, l'algorithme s'arrêtant au premier niveau de variation (la bulle externe), masquant ainsi les événements internes. Cette interprétation nous a été confirmée par les auteurs d'`INVPG annot` : bien que `vg deconstruct` puisse signaler la présence d'une région complexe inversée, il ne peut actuellement pas détailler son contenu exact. Ces discussions ont mis en lumière la nécessité de mettre au point un outil plus approprié que `vg deconstruct` afin de détecter correctement les variants structuraux imbriqués.

E-MSpangepop) Analyse comparative des outils de construction de graphes

À ce jour, deux outils principaux permettent de produire des PVG à partir de grands génomes eucaryotes : `PGGB` [7] et `Minigraph-Cactus (MC)` [8]. `PGGB` adopte une approche sans référence : il réalise un alignement tous-contre-tous avec `wfmash` à partir duquel `seqwish` construit le graphe exact. Une dernière étape, assurée par `smoothxg`, permet d'affiner la topologie pour réduire sa complexité via un alignement d'ordre partiel (POA). `MC` adopte lui une approche guidée par référence. Elle combine `Minigraph` pour construire un squelette de variants structuraux par alignement itératif sur la référence et `Cactus` pour aligner finement les séquences et intégrer les petits variants. Ces différences d'approche conduisent à des topologies de graphes distinctes pour un même jeu de données. De plus, bien que largement utilisés, ces outils sont encore en développement actif, ce qui nuit à leur stabilité et donc à la qualité de leurs résultats. Leur évaluation reste toutefois difficile en raison de l'absence de jeux de données de référence. Ainsi, l'évaluation de ces outils repose habituellement sur la détection des variants, une méthode indirecte et biaisée, qui ne permet pas de juger de la fidélité topologique absolue du graphe. Pour pallier ce manque de standard de référence, nous avons utilisé `MSpangepop` pour générer des graphes topologiquement « parfaits », dérivés directement de la généalogie simulée, afin de les comparer aux reconstructions effectuées par `PGGB` et `MC`.

Méthodologie de comparaison

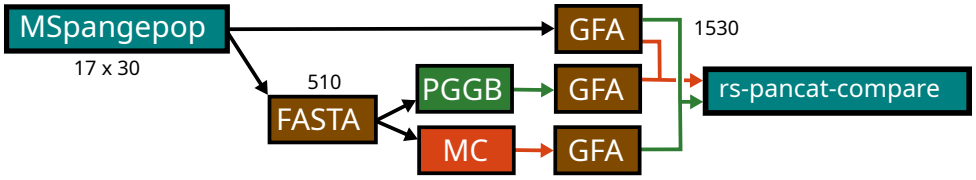

FIGURE III.9 – **Plan expérimental de *benchmarking*** : simulation, reconstruction et calcul de la distance d'édition.

Mon analyse repose sur le plan expérimental suivant (fig. III.9) :

1. **Simulation des graphes :** Génération de 510 graphes avec `MSpangepop`, basés sur le génome de *Staphylococcus aureus*, répartis en **17 modalités expérimentales** (30 réplicats chacune) faisant varier diversité génétique, types de variants majoritaires et nombre d'haplotypes inclus dans le graphe ;
2. **Reconstruction :** À partir des séquences FASTA simulées, reconstruction des graphes via `PGGB` et `MC` (1 020 graphes reconstruits).

3. **Comparaison :** Évaluation de la **Distance d'Édition de Graphe** (GED, *Graph Edit Distance*) entre les graphes simulés (référence) et les graphes reconstruits à l'aide de l'outil `rs-pancat-compare` [29]. Cette métrique quantifie le nombre minimal d'opérations élémentaires (insertion, suppression ou substitution de nœuds/arêtes) nécessaires pour transformer un graphe en un autre.

Cette analyse a nécessité environ **4 200 heures CPU** de calcul sur un HPCC, soit un équivalent carbone approximatif d'un trajet de 70 km en voiture thermique (évaluation par le HPCC Génotoul [20]).

Impact de la diversité génétique (θ)

L'impact de la diversité génétique populationnelle a été évalué pour des valeurs de $\theta \in \{0, 2 \cdot 10^{-5}, 2 \cdot 10^{-4}, 2 \cdot 10^{-3}, 1 \cdot 10^{-2}\}$. Comme anticipé, la distance d'édition de graphe (GED) montre une corrélation positive avec θ (fig. III.10), reflétant la difficulté croissante des algorithmes à déterminer les homologies de séquences dans un contexte de forte divergence.

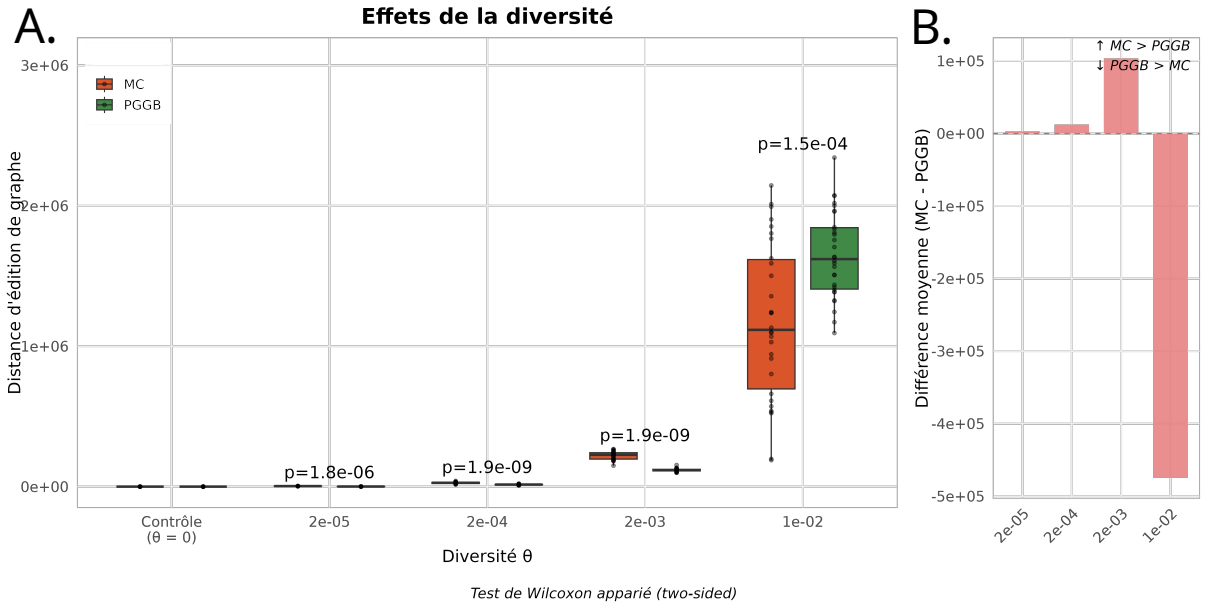

FIGURE III.10 – **Impact de la diversité génétique (θ) sur la qualité de reconstruction des graphes. A** : Distribution de la GED pour `MC` (orange) et `PGGB` (vert) en fonction de θ. La différence entre les deux outils a été testée à l'aide d'un test de Wilcoxon apparié (bilatéral). **B** : Différence moyenne de GED entre les deux méthodes ($\Delta = \text{MC} - \text{PGGB}$). Une valeur positive indique une meilleure performance de `PGGB`, tandis qu'une valeur négative indique une meilleure performance de `MC`.

Les différences observées entre `PGGB` et `MC` sont toujours significatives (test de Wilcoxon apparié, $p < 0.05$) mais l'analyse révèle deux régimes distincts (fig. III.10.A). Pour des valeurs de diversité faibles à modérées ($\theta \leq 2 \cdot 10^{-3}$), la distance d'édition reste limitée avec une augmentation progressive en fonction de θ et des performances marginalement meilleures pour `PGGB` (fig. III.10.B). Une rupture de tendance est observée au seuil de $\theta = 10^{-2}$, où la distance d'édition augmente brusquement de deux ordres de grandeur pour les deux méthodes. Ce décrochage marque probablement la limite des heuristiques d'alignement face à la saturation des mutations. Il est intéressant de noter que `MC` a de meilleures performances pour cette forte valeur de θ . Cet avantage est toutefois atténué

par une **très forte variance**, suggérant que, bien que l'approche guidée par référence de `MC` offre un meilleur ancrage que l'approche tous-contre-tous de `PGGB`, sa stabilité de reconstruction reste précaire dans ces conditions extrêmes.

Sensibilité aux types de variants

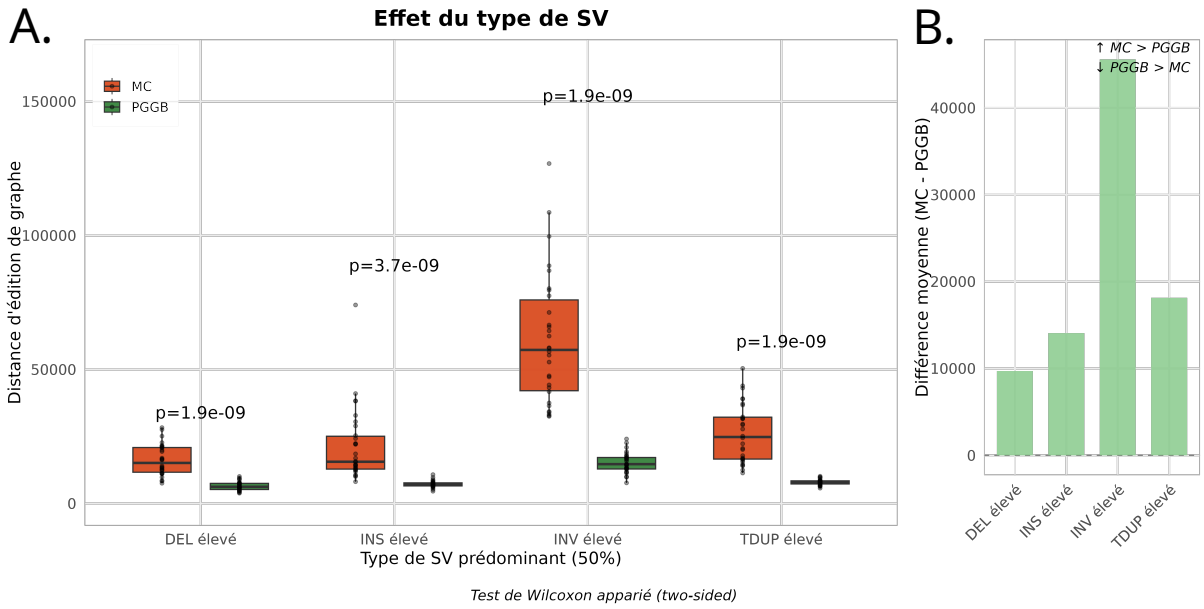

FIGURE III.11 – **Comparaison de la distance d'édition selon le type de variant majoritaire.** Pour chaque modalité, la classe de variants dominante représente 50% des variants structurels (SV). Les conventions graphiques (couleurs, tests statistiques et calcul du différentiel Δ) sont identiques à celles de la Figure III.10.

Nous avons ensuite testé l'impact des différents types de variants en enrichissant à tour de rôle leurs proportions respectives à 50% du total des variants simulés. Dans ce contexte, `PGGB` montre de meilleurs résultats et une variabilité bien plus faible entre types de variants (fig. III.11). Cette analyse révèle également que les deux outils ont plus de difficultés pour reconstruire les **inversions** et les **duplications en tandem** que les autres variants. Cette limitation s'explique par la nature topologique commune de ces deux classes. Contrairement aux insertions ou délétions, les duplications et les inversions impliquent la réutilisation de nœuds existants dans le graphe, soit par un passage répété pour une duplication (boucle), soit par un parcours en sens inverse pour une inversion. Or, si la paramétrisation (e.g. une fenêtre glissante trop courte) et/ou les limites des aligneurs empêchent l'évaluation correcte de l'homologie allélique, tout ou partie de l'allèle alternatif est interprété comme séquence non homologue et est encodé sous forme d'une **insertion factice**. Cela empêche un second passage des chemins par les nœuds déjà existants. Cette linéarisation erronée augmente artificiellement la complexité du graphe et donc la distance d'édition (fig. III.12).

Pour vérifier cette hypothèse de linéarisation, nous avons comptabilisé le nombre total de nœuds générés par chaque méthode sur l'ensemble des 510 graphes. Les résultats confirment notre interprétation : tandis que les graphes simulés contiennent 4 582 048 nœuds, ceux produits par `PGGB` en comptent 4 822 568, soit une augmentation modeste de 5%. En revanche, `MC` génère 10 997 911 nœuds, représentant une inflation de 140% par rapport à la référence simulée. Cette différence substantielle reflète directement la

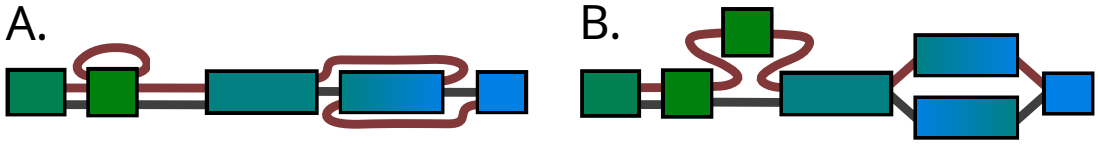

FIGURE III.12 – **Illustration de la linéarisation erronée des variants complexes. (A)** Graphe simulé : les variants (duplication en tandem et inversion) sont correctement encodés par une réutilisation des nœuds existants, les deux chemins (bleu et rouge) partageant les segments homologues. **(B)** Graphe reconstruit : les mêmes variants sont encodés par la création de nœuds supplémentaires, dupliquant une portion de séquence déjà présente dans le graphe. Bien que les séquences génétiques parcourues soient identiques dans les deux cas, la topologie du graphe reconstruit (B) est artificiellement complexifiée.

tendance de `MC` à linéariser les variants topologiquement complexes : chaque inversion ou duplication incorrectement résolue se traduit par la création de nœuds redondants encodant une insertion factice plutôt que par la réutilisation des nœuds existants (fig. III.12). Ces observations corroborent la supériorité de `PGGB` pour la reconstruction fidèle des structures non linéaires.

Effet conjoint de la diversité et de la taille d'échantillon

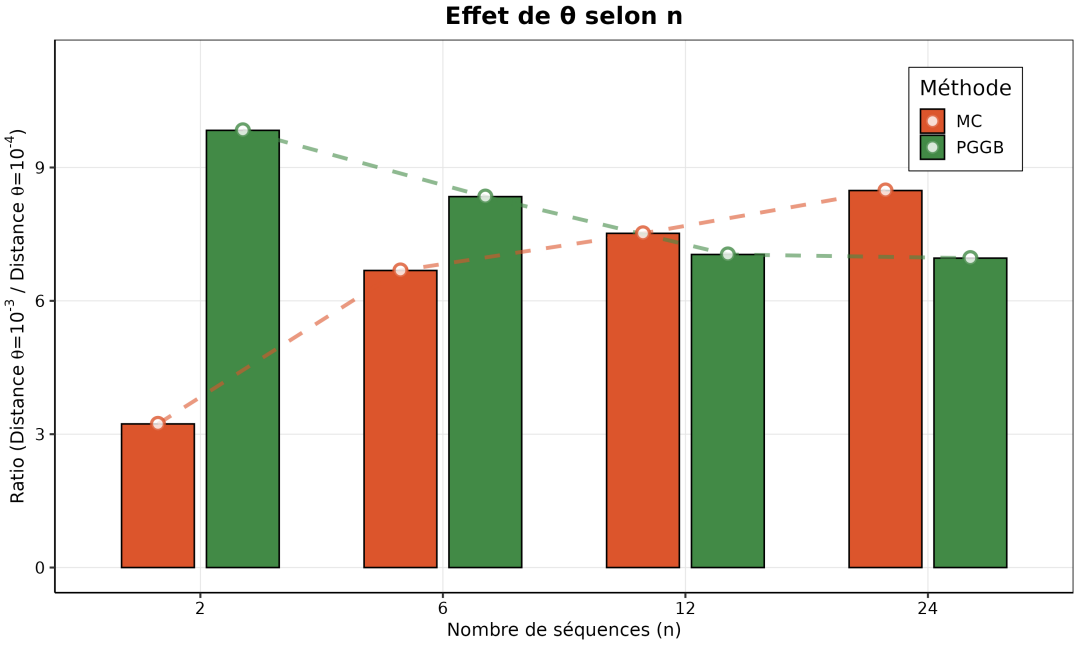

FIGURE III.13 – **Effet conjoint de la taille de l'échantillon (n) et de la diversité (θ) sur la qualité de reconstruction.** La hauteur des barres représente le ratio $R_\theta(n)$, défini comme le rapport entre la distance d'édition à haute diversité ($\theta = 10^{-3}$) et celle à basse diversité ($\theta = 10^{-4}$), pour `MC` (orange) et `PGGB` (vert). Un ratio élevé indique une sensibilité accrue de la méthode à l'augmentation de la diversité génétique. Les courbes en pointillés illustrent les tendances respectives de `PGGB` et `MC`.

Enfin, nous avons analysé l'interaction entre la diversité θ et le nombre d'individus n pour $(\theta, n) \in \{10^{-3}, 10^{-4}\} \times \{2, 6, 12, 24\}$. La distance d'édition augmentant mécaniquement avec le nombre d'haplotypes n, nous avons introduit un ratio normalisé afin d'isoler l'effet de la diversité à n fixé :

$$R_\theta(n) = \frac{\mathrm{GED}(n, \theta_{\text{haut}} = 10^{-3})}{\mathrm{GED}(n, \theta_{\text{bas}} = 10^{-4})}$$

Nous observons que ce ratio se stabilise entre 7 et 9 lorsque n augmente, indiquant que l'effet relatif de la diversité devient indépendant de la taille de l'échantillon (fig. III.13). Ainsi, une augmentation d'un facteur 10 de la diversité (entre θ_{bas} et θ_{haut}) entraîne une augmentation de distance d'édition d'un facteur plus faible, de l'ordre de 7 à 9. Cette relation met en évidence une réponse forte mais non strictement proportionnelle de la complexité du graphe à la diversité génétique. L'effet de la taille de l'échantillon n révèle cependant des comportements différents entre `PGGB` et `MC`. On s'attendrait à ce que $R_\theta(n)$ soit constant pour chaque outil si la distance d'édition était strictement indépendante de la taille de l'échantillon. Or, nous observons des dynamiques opposées : pour `PGGB`, l'impact relatif de la diversité diminue lorsque n augmente, tandis qu'il augmente pour `MC`. Cette divergence illustre les différences algorithmiques fondamentales entre les deux outils. `PGGB`, grâce à son approche tous-contre-tous, bénéficie de l'augmentation du nombre de génomes : la redondance d'information permet de consolider les alignements et de générer un consensus plus robuste, atténuant l'effet de la diversité. À l'inverse, l'approche itérative de `MC` accumule séquentiellement la complexité, chaque génome ajouté dans un contexte de forte diversité augmentant les ambiguïtés du squelette existant et donc la fréquence des erreurs de reconstruction.

F-MSpangepop) Analyse critique et positionnement

L'absence de jeux de données de référence constitue un verrou méthodologique majeur pour le développement d'algorithmes de pangénomique. Les résultats préliminaires observés avec `MSpangepop` s'alignent avec la logique interne des algorithmes testés, validant la cohérence de notre approche de *benchmarking*. Ce besoin de validation est confirmé par l'émergence récente de travaux similaires. Un *workflow* publié en décembre 2025 par Kopalli *et al.* [30] poursuit le même objectif d'évaluation des outils. Toutefois, cette approche présente des limitations significatives que `MSpangepop` a précisément été conçu pour surmonter : absence de modélisation explicite de la recombinaison, spectre restreint de variants structuraux (notamment l'absence d'inversions), et surtout une dépendance aux outils de construction de graphes existants pour générer le PVG de référence. Cet écueil limite la portée de leurs conclusions puisqu'il devient impossible de déterminer si les différences observées entre le graphe de référence et les PVG reconstruits proviennent d'erreurs dans la référence ou dans les PVG reconstruits. Cette publication met ainsi en évidence l'intérêt d'approches alternatives fondées sur des simulations démo-génétiques et sur la construction directe de graphes de référence, telles que celles mises en œuvre dans `MSpangepop`.

Notre comparaison de `PGGB` et `MC` grâce à une référence connue (i.e. simulée), nous a permis de mieux appréhender le comportement de ces deux outils. L'approche sans référence de `PGGB` est plus pertinente pour les échantillons avec une forte diversité génétique (tant que θ n'atteint pas des valeurs extrêmes). En effet, cet outil apparaît structurellement plus robuste face à la complexité topologique et bénéficie de l'ajout de génomes (effet consensus), tandis que `MC` montre une sensibilité accrue à l'accumulation séquentielle d'erreurs lorsque la taille de l'échantillon augmente. Cependant, notre analyse pointe une faiblesse commune aux deux approches : la difficulté à résoudre les variants topologiquement complexes comme les inversions et les duplications. Nos résultats suggèrent que les algorithmes actuels tendent à linéariser ces événements (les transformant en insertions factices) lorsqu'ils échouent à établir l'homologie exacte. Ce constat met en lumière un paradoxe actuel du domaine : bien que nous construisions des objets non-linéaires

(graphes), les méthodes de détection de variants et d'analyse (souvent basées sur la projection linéaire ou la décomposition locale) peinent encore à capturer la complexité native des réarrangements génomiques imbriqués.

Bien que `MSpangepop` marque une avancée, notre analyse repose sur l'hypothèse forte que le graphe généré constitue une représentation « parfaite » du pangénome. Cette hypothèse, bien que raisonnable dans notre cadre contrôlé, demeure partiellement arbitraire. En effet, la pangénomique ne dispose pas encore de standard de représentation unique : pour un même jeu de données, plusieurs topologies de graphes peuvent être considérées comme biologiquement valides selon les choix de modélisation (granularité des nœuds, traitement des régions répétées). Un des atouts de `MSpangepop` est qu'en cas de fixation de standards de représentation, son implantation se basant sur des objets permettra de les adopter aisément. En attendant, cette absence de consensus complique l'interprétation des distances d'édition en termes absolus. À ce titre, la **Distance d'Édition de Graphe** (GED), bien qu'informative, ne caractérise pas finement les différences topologiques locales. L'intégration future de métriques complémentaires, telles que le **ratio de divergence** mesurant la proportion de points de jonction (*breakpoints*) non partagés entre graphes simulé et reconstruit permettrait d'affiner l'évaluation de la qualité des reconstructions sur une échelle normalisée.

Une limite de `MSpangepop`, plus difficile à surmonter, réside dans le mécanisme d'insertion des variants qui reste contraint par le formalisme des modèles de coalescence avec recombinaison (`msprime`) sur lesquels se basent actuellement nos simulations démogénétiques. La définition des locus (blocs de recombinaison) n'est pas un processus itératif au cours des générations mais une étape préliminaire au calcul des généalogies. Ces blocs étant figés, les variants structuraux (délétions, inversions) ne peuvent en pratique pas chevaucher plusieurs locus à un instant t bien que cela soit biologiquement possible. Dans la version actuelle, la portée de ces variants est donc contrainte à l'intervalle $[i, j]$ du locus considéré qui peut être très court en cas de ρ très fort (e.g. comme observé chez *Drosophila melanogaster*, *Mytilus sp.*, *Heliconius sp.* ou *Quercus sp.*).

Finalement, au regard de la complexité des algorithmes et des tests de performance réalisés, nous estimons que la simulation de grands PVG, y compris pour des espèces à grands génomes telles que le maïs, demeure accessible sur des HPCC académiques, dans la limite d'un millier d'individus par graphe. Au-delà de ce seuil, une refonte de l'architecture, ou des stratégies de sous-échantillonnage, seront vraisemblablement requises pour maintenir des temps de calcul acceptables.

IV | Conclusions et perspectives

L'étude des variations structurales constitue un enjeu majeur pour comprendre les bases génomiques de l'adaptation des espèces à leur environnement, particulièrement chez les organismes non modèles comme les arbres forestiers. C'est dans ce cadre que s'inscrit mon alternance, qui avait pour objectif de lever deux verrous méthodologiques : la production automatisée d'assemblages génomiques de haute qualité et l'absence de données de référence permettant d'évaluer objectivement les outils de construction de pangénomes. Afin que les solutions développées soient non seulement performantes mais également réutilisables et comparables, le projet s'est inscrit dans une démarche de science ouverte, en accord avec les principes FAIR. Les deux outils développés sont ainsi publiquement disponibles sur la `ForgeINRAE` et GitHub, et ont été entièrement conteneurisés et documentés.

Le premier axe de mon travail a porté sur la modernisation et la fiabilisation du *workflow* d'assemblage `asm4pg`. La refactorisation majeure du code, réduisant de 58% le nombre de règles Snakemake, a participé à la création d'un outil de production robuste et maintenable. L'intégration des standards actuels, notamment `Hifiasm` et `YaHS` pour le scaffolding Hi-C, a abouti à une chaîne de traitement capable de produire des assemblages de qualité chromosomique avec une intervention humaine minimale, garantissant ainsi l'alimentation des projets de pangénomique en données de haute qualité. Au sein du projet Pangenoak, `asm4pg` a permis l'assemblage de 27 haplotypes issus de diverses espèces de chênes blancs européens, incluant *Quercus robur*, *Q. petraea* et *Q. pubescens*. La qualité des assemblages obtenus a été validée par les métriques de contiguïté et de complétude par rapport aux génomes déjà publiés. L'outil est également employé par d'autres groupes à INRAE (UMR BFP, AGAP), notamment pour des génomes d'arbres fruitiers et la vigne, ainsi que par des membres du CIRAD pour des champignons haploïdes. Toutefois, le *workflow* demeure limité aux génomes haploïdes et diploïdes, excluant pour l'heure les organismes polyploïdes dont le traitement nécessiterait des adaptations substantielles. `Hifiasm` ayant été initialement conçu pour les génomes humains, sa capacité à produire des assemblages polyploïdes repose actuellement sur des développements *ad hoc* plutôt que sur une fonctionnalité intégrée. L'évolution rapide de cet assembleur laisse néanmoins espérer une simplification future de ce traitement, qui faciliterait son intégration dans notre *workflow*. Au-delà de cette limitation, les perspectives d'évolution sont nombreuses : l'intégration de modules dédiés à l'identification des centromères et des génomes organellaires (chloroplaste, mitochondrie), l'ajout d'étapes de polissage supplémentaires via des outils tels que `Verkko-fillet`, ainsi que le développement d'une suite de tests automatisés, encore réalisés manuellement malgré la densification croissante des cas d'usage, constituent autant de pistes d'amélioration. La maintenance des versions les plus récentes des outils, en particulier `Hifiasm`, demeurera une priorité, de même que la rédaction d'un article scientifique visant à entériner une version stable et à permettre aux utilisateurs de citer formellement l'outil.

Le second axe a abouti au développement *de novo* du simulateur `MSpangepop`. Cet outil répond à un besoin critique de la communauté pangénomique : fournir une « vérité terrain » indépendante des biais d'assemblage. En couplant génétique des populations

et théorie des graphes, `MSpangepop` permet désormais de générer des pangénomes synthétiques réalistes respectant un cahier des charges strict : (i) simulation de scénarios démographiques complexes via l'intégration native de `msprime`, (ii) modélisation fine de la recombinaison grâce à l'exploitation de la structure de *tree-sequence*, (iii) génération de variations structurales complexes, notamment les variants emboîtés, grâce à une approche algorithmique de « Traversée » chronologique, (iv) gestion rigoureuse des topologies de graphes bi-orientés via une bibliothèque d'objets Python dédiée, (v) et export direct d'un fichier GFA « parfait » utilisable immédiatement pour le *benchmarking*. L'application concrète à la comparaison `PGGB` / `MC` a notamment permis de montrer la plus grande robustesse du premier face à des valeurs de θ croissantes. Ces résultats confirment qu'il est préférable de s'appuyer sur une référence dont la topologie est mathématiquement maîtrisée afin d'évaluer les outils de construction de graphes. `MSpangepop` est désormais disponible en version `v0.1.0`. Les nombreuses communications menées auprès d'utilisateurs potentiels, notamment lors du Symposium Pangénomique des JOBIM, au sein du réseau GET-A-PAN, du groupe de travail dédié à INRAE et de l'ANR PanQuest, ont suscité un intérêt marqué pour le simulateur. Plusieurs équipes l'utilisent déjà ou ont exprimé leur intention de l'employer pour évaluer leurs propres outils, et leurs retours d'expérience ont déjà contribué à enrichir `MSpangepop`. Les discussions avec les développeurs d'`INVPG annot`, de l'équipe INRIA GenScale, ont notamment permis d'identifier des limitations dans la détection des variants emboîtés par `vg deconstruct`, ouvrant la voie à de futures améliorations méthodologiques. Ces échanges avec GenScale ont permis d'initier une collaboration portant sur la construction de métriques d'évaluation de la qualité des graphes de pangénomes. L'architecture actuelle de `MSpangepop` présente cependant une limitation intrinsèque : lorsque le nombre d'individus simulés devient élevé, la complexité algorithmique explosera et impliquera le développement de nouvelles stratégies, notamment via la programmation dynamique, afin de garantir des temps de calcul raisonnables. Plusieurs axes d'amélioration ont par ailleurs été identifiés pour les versions futures : la gestion des variants chevauchant les points de recombinaison, actuellement contraints aux frontières des locus, permettrait de modéliser des réarrangements plus réalistes ; l'intégration d'une carte de mutation hétérogène le long du génome améliorerait le réalisme biologique en simulant régions conservées (e.g. régions codantes) et points chauds de mutations (e.g. télomères, centromères) ; la prise en compte de bibliothèques d'éléments transposables pour générer des insertions permettrait de simuler la dynamique de ces SV largement étudiés ; enfin, l'ajout de translocations inter-chromosomiques élargirait le spectre des réarrangements possibles, les chromosomes étant pour l'instant traités de manière indépendante.

Sur le plan personnel, cette alternance a été l'occasion de découvrir et d'approfondir deux domaines qui ont suscité chez moi un vif intérêt : l'algorithmique appliquée aux structures de données complexes et la génétique des populations. Le développement de `MSpangepop` m'a confronté à des défis stimulants, tels que la manipulation de graphes bi-orientés ou la modélisation du processus de coalescence, me confirmant dans ma volonté de poursuivre en thèse dans cette voie. Cette expérience m'a également permis de développer des compétences en gestion de projets sur infrastructures HPCC. Les outils développés constituent aujourd'hui un socle solide pour le projet Pangenoak et, plus largement, une contribution utile pour la communauté travaillant sur les variations structurales.

Références bibliographiques et sitographiques

1. Bibliographie

[1] Catlin, N.S., Agha, H.I., Platts, A.E., Munasinghe, M., Hirsch, C.N., Josephs, E.B. : Structural variants contribute to phenotypic variation in maize. Molecular Ecology (2025). doi :10.1111/mec.17662

[2] Chikhi, R., Limasset, A., Medvedev, P. : Compacting de bruijn graphs from sequencing data quickly and in low memory. Bioinformatics **32**(12), 201–208 (2016). doi :10.1093/bioinformatics/btw279

[3] Groza, C., Schwendinger-Schreck, C., Cheung, W.A., Farrow, E.G., Thiffault, I., Lake, J., Rizzo, W.B., Evrony, G., Curran, T., Bourque, G., Pastinen, T. : Pangenome graphs improve the analysis of structural variants in rare genetic diseases. Nature Communications **15**(1) (2024). doi :10.1038/s41467-024-44980-2

[4] Moustakli, E., Christopoulos, P., Potiris, A., Zikopoulos, A., Mavrogianni, D., Karampas, G., Kathopoulis, N., Anagnostaki, I., Domali, E., Tzallas, A.T., Drakakis, P., Stavros, S. : Long-read sequencing and structural variant detection : Unlocking the hidden genome in rare genetic disorders. Diagnostics **15**(14), 1803 (2025). doi :10.3390/diagnostics15141803

[5] Cheng, H., Concepcion, G.T., Feng, X., Zhang, H., Li, H. : Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm. Nature Methods **18**(2), 170–175 (2021). doi :10.1038/s41592-020-01056-5

[6] Oliver, K.R., McComb, J.A., Greene, W.K. : Transposable elements : Powerful contributors to angiosperm evolution and diversity. Genome Biology and Evolution **5**(10), 1886–1901 (2013). doi :10.1093/gbe/evt141

[7] Garrison, E., Guarracino, A., Heumos, S., Villani, F., Bao, Z., Tattini, L., Hagmann, J., Vorbrugg, S., Marco-Sola, S., Kubica, C., Ashbrook, D.G., Thorell, K., Rusholme-Pilcher, R.L., Liti, G., Rudbeck, E., Golicz, A.A., Nahnsen, S., Yang, Z., Mwaniki, M.N., Nobrega, F.L., Wu, Y., Chen, H., de Ligt, J., Sudmant, P.H., Huang, S., Weigel, D., Soranzo, N., Colonna, V., Williams, R.W., Prins, P. : Building pangenome graphs. Nature Methods **21**(11), 2008–2012 (2024). doi :10.1038/s41592-024-02430-3

[8] Hickey, G., Monlong, J., Ebler, J., Novak, A.M., Eizenga, J.M., Gao, Y., Abel, H.J., Antonacci-Fulton, L.L., Asri, M., Baid, G., Baker, C.A., Belyaeva, A., Billis, K., Bourque, G., Buonaiuto, S., Carroll, A., Chaisson, M.J.P., Chang, P.-C., Chang, X.H., Cheng, H., Chu, J., Cody, S., Colonna, V., Cook, D.E., Cook-Deegan, R.M., Cornejo, O.E., Diekhans, M., Doerr, D., Ebert, P., Ebler, J., Eichler, E.E., Fairley, S., Fedrigo, O., Felsenfeld, A.L., Feng, X., Fischer, C., Flicek, P., Formenti, G., Frankish, A., Fulton, R.S., Garg, S., Garrison, E., Garrison, C.G., Green, R.E., Groza, C., Guarracino, A., Haggerty, L., Hall, I.M., Harvey, W.T., Haukness, M., Haussler, D., Heumos, S., Hoekzema, K., Hourlier, T., Howe, K., Jain, M., Jarvis, E.D., Ji, H.P., Kenny, E.E., Koenig, B.A., Kolesnikov, A., Korbel, J.O., Kordosky, J., Koren, S., Lee, H., Lewis, A.P., Liao, W.-W., Lu, S., Lu, T.-Y., Lucas, J.K., Magalhães, H., Marco-Sola, S., Marijon, P., Markello, C., Marschall, T., Martin, F.J., McCartney, A.,

McDaniel, J., Miga, K.H., Mitchell, M.W., Mountcastle, J., Munson, K.M., Mwaniki, M.N., Nattestad, M., Nurk, S., Olsen, H.E., Olson, N.D., Pesout, T., Phillippy, A.M., Popejoy, A.B., Porubsky, D., Prins, P., Puiu, D., Rautiainen, M., Regier, A.A., Rhie, A., Sacco, S., Sanders, A.D., Schneider, V.A., Schultz, B.I., Shafin, K., Sibbesen, J.A., Sirén, J., Smith, M.W., Sofia, H.J., Tayoun, A.N.A., Thibaud-Nissen, F., Tomlinson, C., Tricomi, F.F., Villani, F., Vollger, M.R., Wagner, J., Walenz, B., Wang, T., Wood, J.M.D., Zimin, A.V., Zook, J.M., Marschall, T., Li, H., Paten, B. : Pangenome graph construction from genome alignments with minigraph-cactus. Nature Biotechnology **42**(4), 663–673 (2023). doi :10.1038/s41587-023-01793-w

[9] Kremer, A., Delcamp, A., Lesur, I., Wagner, S., Rellstab, C., Guichoux, E., Leroy, T. : Whole-genome screening for near-diagnostic genetic markers for four western european white oak species identification. Annals of Forest Science **81**(1) (2024). doi :10.1186/s13595-024-01236-9

[10] Saleh, D., Chen, J., Leplé, J.-C., Leroy, T., Truffaut, L., Dencausse, B., Lalanne, C., Labadie, K., Lesur, I., Bert, D., Lagane, F., Morneau, F., Aury, J.-M., Plomion, C., Lascoux, M., Kremer, A. : Genome-wide evolutionary response of european oaks during the anthropocene. Evolution Letters **6**(1), 4–20 (2022). doi :10.1002/evl3.269

[11] Davenport, C., Chevreau, J., Zhang, Y., Siegfried, Piat, L., Glombik, M., Deng, C., Diesh, C., Dainat, J., Gagalova, K., Majidian, S. : colindaven/awesome-pangenomes : v1 Release. Zenodo (2026). doi :10.5281/zenodo.18208385. https://doi.org/10.5281/zenodo.18208385

[12] Molder, F., Jablonski, K.P., Letcher, B., Hall, M.B., van Dyken, P.C., Tomkins-Tinch, C.H., Sochat, V., Forster, J., Vieira, F.G., Meesters, C., Lee, S., Twardziok, S.O., Kanitz, A., VanCampen, J., Malladi, V., Wilm, A., Holtgrewe, M., Rahmann, S., Nahnsen, S., Koster, J. : Sustainable data analysis with snakemake. F1000Research **10**, 33 (2025). doi :10.12688/f1000research.29032.3

[13] Kurtzer, G.M., Cclerget, Bauer, M., Kaneshiro, I., Trudgian, D., Godlove, D. : hpcng/singularity : Singularity 3.7.3. Zenodo (2021). doi :10.5281/ZENODO.1310023. https://zenodo.org/record/1310023

[14] Alonge, M., Lebeigle, L., Kirsche, M., Jenike, K., Ou, S., Aganezov, S., Wang, X., Lippman, Z.B., Schatz, M.C., Soyk, S. : Automated assembly scaffolding using ragtag elevates a new tomato system for high-throughput genome editing. Genome Biology **23**(1) (2022). doi :10.1186/s13059-022-02823-7

[15] Blaxter, M., Mieszkowska, N., Di Palma, F., Holland, P., Durbin, R., Richards, T., Berriman, M., Kersey, P., Hollingsworth, P., Wilson, W., Twyford, A., Gaya, E., Lawniczak, M., Lewis, O., Broad, G., Howe, K., Hart, M., Flicek, P., Barnes, I. : Sequence locally, think globally : The darwin tree of life project. Proceedings of the National Academy of Sciences **119**(4) (2022). doi :10.1073/pnas.2115642118

[16] Goate, Z. : The genome sequence of the two-spot ladybird, adalia bipunctata (linnaeus, 1758). Wellcome Open Research **7**, 288 (2022). doi :10.12688/wellcomeopenres.18610.1

[17] Kingman, J.F.C. : The coalescent. Stochastic Processes and their Applications **13**(3), 235–248 (1982). doi :10.1016/0304-4149(82)90011-4

[18] Deng, Y., Nielsen, R., Song, Y.S. : Robust and accurate bayesian inference of genome-wide genealogies for hundreds of genomes. Nature Genetics **57**(9), 2124–2135 (2025). doi :10.1038/s41588-025-02317-9

[19] Tsambos, G., Kelleher, J., Ralph, P., Leslie, S., Vukcevic, D. : link-ancestors : fast simulation of local ancestry with tree sequence software. Bioinformatics Advances **3**(1) (2023). doi :10.1093/bioadv/vbad163

[20] Baumdicker, F., Bisschop, G., Goldstein, D., Gower, G., Ragsdale, A.P., Tsambos, G., Zhu, S., Eldon, B., Ellerman, E.C., Galloway, J.G., Gladstein, A.L., Gorjanc, G., Guo, B., Jeffery, B., Kretzschumar, W.W., Lohse, K., Matschiner, M., Nelson, D., Pope, N.S., Quinto-Cortés, C.D., Rodrigues, M.F., Saunack, K., Sellinger, T., Thornton, K., van Kemenade, H., Wohns, A.W., Wong, Y., Gravel, S., Kern, A.D., Koskela, J., Ralph, P.L., Kelleher, J. : Efficient ancestry and mutation simulation with msprime 1.0. Genetics **220**(3) (2021). doi :10.1093/genetics/iyab229

[21] Paten, B., Eizenga, J.M., Rosen, Y.M., Novak, A.M., Garrison, E., Hickey, G. : Superbubbles, ultrabubbles, and cacti. Journal of Computational Biology **25**(7), 649–663 (2018). doi :10.1089/cmb.2017.0251

[22] Gillaspy, A.F., Worrell, V., Orvis, J., Roe, B.A., Dyer, D.W., Iandolo, J.J. : The Staphylococcus aureus NCTC 8325 Genome, pp. 381–412. ASM Press, ??? (2014). doi :10.1128/9781555816513.ch32. http://dx.doi.org/10.1128/9781555816513.ch32

[23] Sudmant, P.H., Rausch, T., Gardner, E.J., Handsaker, R.E., Abyzov, A., Huddleston, J., Zhang, Y., Ye, K., Jun, G., Hsi-Yang Fritz, M., Konkel, M.K., Malhotra, A., Stutz, A.M., Shi, X., Paolo Casale, F., Chen, J., Hormozdiari, F., Dayama, G., Chen, K., Malig, M., Chaisson, M.J.P., Walter, K., Meiers, S., Kashin, S., Garrison, E., Auton, A., Lam, H.Y.K., Jasmine Mu, X., Alkan, C., Antaki, D., Bae, T., Cerveira, E., Chines, P., Chong, Z., Clarke, L., Dal, E., Ding, L., Emery, S., Fan, X., Gujral, M., Kahveci, F., Kidd, J.M., Kong, Y., Lameijer, E.-W., McCarthy, S., Flicek, P., Gibbs, R.A., Marth, G., Mason, C.E., Menelaou, A., Muzny, D.M., Nelson, B.J., Noor, A., Parrish, N.F., Pendleton, M., Quitadamo, A., Raeder, B., Schadt, E.E., Romanovitch, M., Schlattl, A., Sebra, R., Shabalin, A.A., Untergasser, A., Walker, J.A., Wang, M., Yu, F., Zhang, C., Zhang, J., Zheng-Bradley, X., Zhou, W., Zichner, T., Sebat, J., Batzer, M.A., McCarroll, S.A., Mills, R.E., Gerstein, M.B., Bashir, A., Stegle, O., Devine, S.E., Lee, C., Eichler, E.E., Korbel, J.O. : An integrated map of structural variation in 2, 504 human genomes. Nature **526**(7571), 75–81 (2015). doi :10.1038/nature15394

[24] Larivière, D., Abueg, L., Brajuka, N., Gallardo-Alba, C., Gruning, B., Ko, B.J., Ostrovsky, A., Palmada-Flores, M., Pickett, B.D., Rabbani, K., Antunes, A., Balacco, J.R., Chaisson, M.J.P., Cheng, H., Collins, J., Couture, M., Denisova, A., Fedrigo, O., Gallo, G.R., Giani, A.M., Gooder, G.M., Horan, K., Jain, N., Johnson, C., Kim, H., Lee, C., Marques-Bonet, T., O'Toole, B., Rhie, A., Secomandi, S., Sozzoni, M., Tilley, T., Uliano-Silva, M., van den Beek, M., Williams, R.W., Waterhouse, R.M., Phillippy, A.M., Jarvis, E.D., Schatz, M.C., Nekrutenko, A., Formenti, G. : Scalable, accessible and reproducible reference genome assembly and evaluation in galaxy. Nature Biotechnology **42**(3), 367–370 (2024). doi :10.1038/s41587-023-02100-3

[25] Kelleher, J., Etheridge, A.M., McVean, G. : Efficient coalescent simulation and genealogical analysis for large sample sizes. PLOS Computational Biology **12**(5), 1004842 (2016). doi :10.1371/journal.pcbi.1004842

[26] Hudson, R.R. : Generating samples under a wright–fisher neutral model of genetic variation. Bioinformatics **18**(2), 337–338 (2002). doi :10.1093/bioinformatics/18.2.337

[27] Romain, S., Dubois, S., Legeai, F., Lemaitre, C. : Investigating the topological motifs of inversions in pangenome graphs. bioRxiv (2025). doi :10.1101/2025.03.14.643331. https://www.biorxiv.org/content/early/2025/03/17/2025.03.14.643331.full.pdf

[28] Garrison, E., Sirén, J., Novak, A.M., Hickey, G., Eizenga, J.M., Dawson, E.T., Jones, W., Garg, S., Markello, C., Lin, M.F., Paten, B., Durbin, R. : Variation graph toolkit improves read mapping by representing genetic variation in the reference. Nature Biotechnology **36**(9), 875–879 (2018). doi :10.1038/nbt.4227

[29] Dubois, S., Zytnicki, M., Lemaitre, C., Faraut, T. : Pairwise graph edit distance characterizes the impact of the construction method on pangenome graphs. Bioinformatics, 291 (2025). doi :10.1093/bioinformatics/btaf291. Accessed 2025-05-14

[30] Kopalli, V., Arslan, K., Morales-Díaz, N., Zanini, S.F., Golicz, A.A. : Toward a standardized framework for pangenome graph evaluation : assessing crop plant pangenome variation graph construction from multiple assemblies. GigaScience **14** (2025). doi :10.1093/gigascience/giaf121

2. Sitographie

1. Christophe Plomion. 03 juin 2025 *UMR BioGeCo*. Consulté le 9 octobre 2025, à l'adresse `https://biogeco.hub.inrae.fr/umr-biogeco/presentation`
2. Marcus Wallenberg Foundation. 2006 *Prix Marcus Wallenberg*. Consulté le 10 décembre 2025, à l'adresse `https://www.mwp.org/%20Previous%20Winners/2006-antoine-kremer-france/`
3. Benjamin Brachi. 03 juin 2025 *Equipe E4E*. Consulté le 9 octobre 2025, à l'adresse `https://biogeco.hub.inrae.fr/equipes/e4e`
4. Wikitionnaire. 29 mars 2024 *Wikitionnaire pangénome | Fondation Wikimédia*. Consulté le 9 octobre 2025, à l'adresse `https://fr.wiktionary.org/wiki/pang%C3%A9nome`
5. Centre de Bioinformatique de Bordeaux. 2025 *CBiB – Bordeaux Bioinformatics Center*. Consulté le 17 décembre 2025, à l'adresse `https://www.cbib.u-bordeaux.fr/`
6. Genotoul Bioinfo, INRAE. avril 2025 *Hardware – genotoul-bioinfo*. Consulté le 17 décembre 2025, à l'adresse `https://bioinfo.genotoul.fr/index.php/resources-2/resources/`
7. Coder Inc. 2025 *code-server : VS Code in the browser*. Consulté le 17 décembre 2025, à l'adresse `https://github.com/coder/code-server`
8. JeBiF, SFBI, Bioinfo-Fr.net. 2025 *Bioinformations*. Consulté le 17 décembre 2025, à l'adresse `https://jebif.fr/liens-utiles/bioinformations/`
9. ResearchRabbit Inc. 2025 *ResearchRabbit : AI Tool for Smarter, Faster Literature Reviews*. Consulté le 17 décembre 2025, à l'adresse `https://www.researchrabbit.ai/`
10. INRAE. 2025 *Forge INRAE – GitLab*. Consulté le 17 décembre 2025, à l'adresse `https://forge.inrae.fr/`
11. SFBI, Comité d'organisation JOBIM 2025. 2025 *JOBIM 2025 – Journées Ouvertes en Biologie, Informatique et Mathématiques*. Consulté le 17 décembre 2025,

à l'adresse https://jobim2025.labri.fr/

12. INRAE. Duvaux *et al.* 10 décembre 2025 *GenomAsm4pg*. Consulté le 17 décembre 2025, à l'adresse https://forge.inrae.fr/asm4pg/GenomAsm4pg
13. INRAE. Duvaux *et al.* 2025 *MSpangepop*. Consulté le 17 décembre 2025, à l'adresse https://forge.inrae.fr/pangepop/MSpangepop
14. Ma petite encyclopédie informatique et arts graphiques, Définition de *exétron* Consulté le 9 janvier 2026, à l'adresse https://ma-petite-encyclopedie.org/accueil?lex_item=ex%C3%A9tron
15. INRAE. Duvaux *et al.* 2024 *GenomAsm4pg (version archivée)*. Consulté le 17 décembre 2025, au commit n°8ab48dd24cd17467d27b5da1e0ad297dcf83775e
16. GFA-spec Community. 2025 *GFA-spec : Graphical Fragment Assembly (GFA) Format Specification*. Consulté le 17 décembre 2025, à l'adresse https://github.com/GFA-spec/GFA-spec
17. Robert C. Holte, Denys Duchier, University of Alberta. s.d. *DSS T26 – Data Structure Course*. Consulté le 17 décembre 2025, à l'adresse https://webdocs.cs.ualberta.ca/~holte/T26/top.realTop.html
18. Darwin Tree of Life Consortium. s.d. *Quercus robur genome project (BioProject PRJEB51283)*. Consulté le 5 janvier 2026, à l'adresse https://db.cngb.org/data_resources/project/PRJEB51283
19. National Institute of Standards and Technology (NIST). s.d. *Genome in a Bottle Consortium*. Consulté le 5 janvier 2026, à l'adresse https://www.nist.gov/programs-projects/genome-bottle
20. Françoise Berthoud *et al* 2020. *Estimation de l'empreinte carbone d'une heure.cœur de calcul* (HPCC Genotoul Bioinfo). Consulté le 6 janvier 2026, à l'adresse https://hal.science/hal-02549565v5

Résumé

Les variants structuraux tels que les insertions, délétions, inversions et duplications constituent une part majeure de la diversité génétique et ont un fort impact sur l'adaptation des espèces à leur environnement. Ces réarrangements restent néanmoins sous-étudiés en raison de leur complexité intrinsèque. L'émergence récente des graphes de variation pangénomiques offre un cadre prometteur pour intégrer l'ensemble des variants au sein d'une représentation unifiée capturant la diversité génomique complète d'un groupe d'individus. Toutefois, le manque de *workflows* d'assemblage clé en main freine l'application de la pangénomique aux espèces non modèles. De plus, l'absence de données de référence à topologie connue empêche toute évaluation objective des outils de construction, créant un paradoxe méthodologique où l'on construit des graphes sans pouvoir en vérifier l'exactitude.

Dans le cadre du projet Pangenoak sur les chênes blancs européens, j'ai modernisé le *workflow* d'assemblage asm4pg afin d'en faciliter l'usage par les biologistes. J'y ai également intégré des étapes de *scaffolding* Hi-C ainsi que des assembleurs de dernière génération, permettant la production de génomes de qualité chromosomique. Afin de répondre au besoin de données de référence réalistes, j'ai développé MSpangepop, un simulateur générant des pangénomes au format GFA en couplant simulations démo-génétiques par coalescence et construction de graphes bi-orientés supportant les variants emboîtés. Son application au *benchmarking* de PGGB et Minigraph-Cactus (MC) met en évidence des réponses contrastées à la diversité génétique : PGGB préserve les topologies complexes lorsque celle-ci est élevée, contrairement à MC. En liant structures de données de coalescence et graphes de variation, MSpangepop est le premier outil permettant des simulations évolutivement réalistes de pangénomes, ouvrant la voie à une exploitation plus fiable des variants structuraux.

Mots-clés : Variation structurale du génome ; Pangénomique ; Coalescence ; Assemblage de génomes ; Graphes de variation

Mots-clés des acquis techniques : Python ; Snakemake ; Conteneurisation ; Calcul haute performance