Vizitig: context-rich exploration of sequencing datasets

Bastien DEGARDINS[1], Charles PAPERMAN[1] and Camille MARCHET[1]

1. Univ. Lille, CNRS, Inria, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France

Corresponding author: camille.marchet@univ-lille.fr

Keywords transcriptomics, pangenomics, de Bruijn graph, visualization, databases

Abstract *Recent advances in k-mer indexing have facilitated the cataloging and rapid querying of planetary-scale genomic data. While these indices excel at high-throughput sequence lookups, they often lack context-rich exploration capabilities and rely on simplistic match-based queries. This gap hinders deeper investigations into variants, regulatory elements, and other features crucial for pangenomic and transcriptomic analyses. We present Vizitig, a novel system that harnesses a de Bruijn graph as the core data structure. By directly encoding overlapping k -mers from both genome and transcriptome data, Vizitig supports the processing of partially or completely unassembled sequences, making it broadly applicable from collections of genomes to eukaryotic RNA-seq. Vizitig integrates k-mer indices into a database framework, providing an intuitive, metadata-aware approach to querying. Users can select candidate regions by specific annotations (e.g., genes, motifs) or sample-specific features (e.g., abundance, presence or absence in annotated gene or a sample), retrieving relevant graph neighborhoods and associated meta-data from extensive datasets.*

Introduction

The rapid advancement of sequencing technologies has ushered in an era of unprecedented genomic data generation, fostering the transition from single-reference genomes to comprehensive collections representing the genetic diversity of entire species, populations, or large cohorts. One prominent type of these collections are pangenomes, aggregates of multiple assembled genomes that provide a nuanced view of genomic variation. Pangenomes are particularly valuable where genetic diversity underpins essential biological functions. For instance, in human genomics, pangenomes enable improved representation of underrepresented populations, thereby enhancing equity in genomic medicine [**?**]. However, the effectiveness of these resources heavily relies on computational frameworks capable of efficiently organizing and analyzing genomic diversity at scale—an ongoing challenge in the field.

Another significant category includes raw sequence indexes. Recent advances in k-mer indexing have facilitated cataloging and rapid querying of planetary-scale genomic data [1][2]. While these indices excel at high-throughput sequence lookups, they often lack the contextual depth necessary for exploratory analyses, relying predominantly on simplistic match-based queries. This limitation impedes deeper investigations into variants, regulatory elements, and other features critical for advanced pangenomic and transcriptomic analyses. Among the computational tools driving the development of raw sequence indexing, de Bruijn graphs have become central to the representation and analysis of large-scale sequence data [3]. These graphs decompose sequences into overlapping k-mers, constructing a topology that represents similarity and variability within and across genomes. Their ability to compactly encode extensive genomic datasets in a scalable way while retaining vari-

ants has made them instrumental for genome assembly [4], variant detection [**?**], and comparative genomics [**?**].

Planetary-scale indexes, such as Logan Search [1], facilitate querying expansive genomic datasets at an unprecedented scale, utilizing indirect representations derived from de Bruijn graphs. Such indexes surpass traditional pangenomics boundaries [5], applicable not only to complete and assembled pangenomic collections but also extending to unassembled, heterogeneous datasets and transcriptomic analyses. Yet, despite their considerable promise, de Bruijn graph-based methods for actionable pangenomics remain underutilized. Previously, with limited datasets, de Bruijn graph approaches leveraged explicit graph topology through graph traversal to detect and analyze genomic variations [6,7]. However, performance constraints have since favored more scalable indexing methods [3]. The current limitation is their reliance on atomic queries—primarily k-mer lookups—that facilitate sequence matching in an alignment-free manner but require significant additional engineering and cross-tool manipulations to translate into biologically meaningful investigations [8,9].

We present **Vizitig**, a novel system that harnesses a de Bruijn graph as the core data structure. By directly encoding overlapping *k*-mers from both genome and transcriptome data, Vizitig supports the processing of partially or completely unassembled sequences, making it broadly applicable from collections of genomes to RNA-seq. Vizitig integrates k-mer indices into a database framework, providing an intuitive, metadata-aware approach to querying. Users can select candidate regions by specific annotations (e.g., genes, exons) or sample-specific features (e.g., abundance, presence or absence in annotated gene or a sample), retrieving relevant graph neighborhoods and associated meta-data from extensive datasets. Key advances in Vizitig are:

- **Efficient exploration:** Only loads genomic contexts of interest, ensuring scalability to large datasets with interactive expansion as needed.
- **Integration of raw sequencing data:** Facilitates augmentation of existing pangenome resources directly with unassembled sequence data or co-integration of raw RNA-seq and reference data.
- **Locus-based queries:** Allows users to identify and explore genomic regions around known biomarkers quickly and without sequence alignment.
- **Trait-based queries:** Supports querying based on metadata traits to discover new genomic regions of interest.
- **Metadata integration:** Enables users to query using gene annotations, sample abundances, or other metadata.
- **Intuitive interface:** Offers a human-readable query language and a feature-complete web interface with the sequences in the graph and metadata.
- **Flexible and accessible design:** Includes command-line and Python APIs; compatibility with common graph analysis tools and diverse computing environment, data export and sharing capabilities in canonical formats (see also Figure 1).

Notably, Vizitig's approach is distinct from graph visualization tools like Bandage [10]. While Bandage focuses on visualizing assembly graphs to facilitate genome assembly debugging, Vizitig integrates comprehensive metadata exploration and query (e.g., annotations, sample provenance, expression abundances) directly within the graph structure. For pangenomics, Panache [11] or PanTools [12] provide interactive exploration focused on gene presence/absence across multiple genomes, and

ODGI [13] offers visualization of graph-based pangenomes with scalability for large datasets, emphasized on structural variations, instead of base level granularity in our case.

Methods

Vizitig is a comprehensive database management system designed to handle sequence graphs, enabling efficient local genomic exploration of multiple samples simultaneously. Vizitig integrating metadata and annotations directly into the graph structure, and exposes them through a visual, user-friendly interface.

Graph model We use a de Bruijn graph to represent DNA and RNA sequencing data. De Bruijn graphs' nodes correspond to strings of size k extracted from all positions in the data – k-mers - and edges link adjacent k-mers with k-1 sequence overlap. Vizitig works with unitig graphs – or compacted de Bruijn graphs [14], a simplified and lossless form of de Bruijn graph that enhance visual clarity. De Bruijn graphs are widely employed in genome assembly and variant detection due to their capability to reconstruct genomic sequences from reads, and to highlight biological variations into distinctive graph patterns, such as bubbles [7]. We employ colored de Bruijn graphs [3] that integrate multiple samples.

Backend Vizitig's backend currently utilizes SQLite [1] to manage unitig-defined graph structures, associating metadata directly with nodes. It incorporates reference sequences by marking nodes containing corresponding k-mers with relevant metadata from DNA/RNA sequences or annotations (GTF/GFF), and supports additional metadata such as color (sample identification) and abundance (occurrence in samples). Efficient retrieval is ensured through indexing, providing logarithmic query time for single k-mers and linear performance for other sequence lengths. A Rust- and Python-based command line interface offers comprehensive backend access for data manipulation. We chose not to rely on analytical graph databases such as Neo4j, as no mature, open-source, embedded solutions currently exist that can run on standard machines. Additionally, such databases are typically not optimized for rapid retrieval of localized graph regions, our primary query focus. To ensure optimal performance, we developed ad hoc indexes tailored for fast localized queries and integrated them with NetworkDisk [2], a disk-based Python module compatible with NetworkX, facilitating large-scale graph manipulation with minimal memory usage by loading only relevant subsets during exploration.

Frontend Vizitig's frontend employs a web-based API connecting to the command line inferface, developed in JavaScript [3], enabling intuitive visualization and interaction with graph data and metadata. It dynamically loads only relevant graph segments for scalable, real-time analyses. This ensures scalability when working with large datasets, as most of the graph sits on the disk while useful bits are loaded. A human-readable domain-specific query language (DSL), supported by a Python-based query assistant, translates user queries into executable NetworkDisk commands, facilitating graph manipulation. Additional DSL-driven filtering and transformation tools allow targeted data extraction and visualization. Vizitig includes automated graph refinement tools, such as the "tip removal"

1. https://www.sqlite.org/
2. https://networkdisk.inria.fr/
3. https://d3js.org/

procedure (`tip hide` action), a common practice in assembly-based analyses [4], to eliminate small erroneous paths typically resulting from sequencing errors. Finally, the system supports data export, enabling easy sharing and downstream analyses.

Queries and actions Vizitig's DSL allows to build queries that can be executed both as a retrieval query as well as actions. The former fetches sequences and meta-data from the database, while the later aims at applying visual transformations on data once it was retrieved. Both can be related to the sequence content (locus based queries) or to the meta-data (trait-based queries), or a combination thereof.

More precisely, the DSL is composed of logical operators (AND, OR and NOT), parenthesis and tokens. Tokens are made of a type keyword, open parenthesis, an optional key and a closing parenthesis. Type keywords correspond to the different types of metadata and are generated based on the annotations files (e.g. `Gene`, `Transcript`, `Exon`). The specific type keyword `Color` allows a second argument `A` that comes with an operator (`<` , `>` , `<=` , `=>` or =) and a value, that allows to query abundances. The keyword `Seq` finds nodes of the graph containing a given sequence of arbitrary size.

Once a subgraph $\mathcal{G}$ is loaded from a retrieval query, visual effects can be superimposed on it.

To determine which nodes and edges of $\mathcal{G}$ will be affected by a visual transformation, a filter is created: a second DSL query is used to select a subgraph of $\mathcal{G}$. Users can create as many filters as needed. Actions associate a given filter with a visual transformation (e.g., hiding nodes, highlighting nodes in a specific color, tagging nodes, adding abundance information, changing node opacity, or labeling nodes). These actions can be stacked to build detailed visualizations of specific subgraph properties. Vizitig has pre-computed filters, such as the tip filter that targets nodes of degree one.

Export and save Vizitig graphs can be exported in various formats (json, gml, BCALM2, svg, ...), and the graph databases and created filters can be shared. Views of a subgraph containing visual transformations can be saved and later reloaded.

Results

For each use case, in addition to command lines, we share the graphs in a vizitig readable format if readers want to explore it themselves on zenodo, DOI: `doi.org/10.5281/zenodo.15148532`.

A typology of variants observed with Vizitig To illustrate genomic variant types detectable by Vizitig, we selected the human gene *CIC* as a case study. We constructed a compacted de Bruijn graph from three RNA-seq samples (SRR8616107, SRR8615242, SRR8615240), enriched with GRCh38 RefSeq transcripts and exon-level annotations from NCBI. Each node was annotated with dataset provenance and k-mer abundance, enabling expression differentiation. Figure 2(a) shows the graph after querying `Gene(CIC)` and expanding connected nodes iteratively. Exons were distinctly colored using metadata annotations via the `highlight` action, with exon junctions highlighted based on transcript metadata. Uncolored nodes indicated novel or unannotated variants or artifacts. Small erroneous paths were removed using a "tip removal" (`tip hide`) action. We highlight various classes of variants

detected by Vizitig. For each class, we provide an theoretical graph example and the corresponding mapping on the human genome:

- Alternative exon splicing: appears as paths excluding one or more complete exons, with endpoints connected to flanking exon sequences (Figure 2(b)).
- Small mutations (indels, mismatches): manifest as alternative paths within an exon region, diverging from but reconnecting to annotated sequences (Figure 2(c)).
- Alternative exon start/stop sites: identified as novel exon-exon junction paths, differing from standard annotated splice junctions (Figure 2(d)).
- In the genomic context, larger variations such as structural variants will create alternative paths in the graph (Figure 5).
- Repeats larger than k, will be collapsed in single regions of the graph.

Locus-based query of a fusion transcript event We further demonstrate Vizitig's utility by investigating a previously documented fusion transcript between the *HM13* and *TPX2* genes, identified through the Cancer Cell Line Encyclopedia (CCLE) and DepMap/Cosmic fusion database [15]. Fusion junction coordinates were obtained from STAR-Fusion [16] alignments provided by DepMap, following guidelines to select events with high reliability. A de Bruijn graph (k=61) was constructed from two control RNA-seq datasets (SRR8615240, SRR8615242) and one dataset known to harbor the fusion event (SRR8616107). Reference transcripts and abundance data were annotated similarly to the previous example. Using the fusion breakpoint sequence as a locus query within Vizitig—capable of querying sequences both shorter and longer than the k-mer size—we identified the exact node corresponding to the fusion junction (Figure 4). This node, not associated with known genes or annotations, remained distinctly uncolored. Application of the `Sashimi` action revealed expression specificity of this fusion event to the fusion dataset (Figure 4 (b)), confirming its sample-specific presence.

Trait-based query revealing a putative insertion in cancer data Search in current colored de Bruijn graph approaches requires a biomarker/known sequence to perform a lookup – which is not mandatory to explore the graph of Vizitig. To illustrate trait-based exploration without prior sequence knowledge, we investigated candidate variants in the *AP3D1* gene using the previously constructed graph. A metadata-driven query (`"Color(SRR8616107,A>200) and Color(SRR8615240,A<10) and Color(SRR8615242,A<10) and Gene(AP3D1)"`) was executed to identify sequences specifically abundant in SRR8616107 compared to other samples, potentially indicative of differential expression or genomic variants. Identified nodes were expanded, and AP3D1 sequences were highlighted. To exclude potential sequencing errors, we created a secondary filter (`"Color(SRR8616107,A<10) and not(Gene(AP3D1))"`) and suppressed these low-abundance, non-gene-specific nodes. A robust candidate variant persisted, showing markedly higher expression in SRR8616107. The alignment of this candidate sequence using UCSC's BLAT [17] tool [4] on the GRCh38 reference revealed a genomic insertion event in exon 22 of the gene.

Pangenome augmentation with long reads We built a pangenome of *C. elegans* using two reference chromosome I from NCBI (Bristol N2 and hawaiian CB4856), and augmented the pangenome with a

4. as of March 2025 on https://genome.ucsc.edu

third strain JU310 sequenced using PacBio long reads and assembled with Canu[18] (`GCA_016989275.1` accession in NCBI). We built graphs with GGCAT (*-k 1023 -s 1*). We show a 10% fraction of chromosome I in Figure 5, where we illustrate our capacity to reveal inter-reference variants as well as variants in the JU310 dataset.

Performance metrics In our example using three human RNA-seq datasets, the Vizitig input files comprises 4.6 GB. The construction was entirely run on a laptop in 16 minutes (wallclock time, 48 minutes CPU time), and using up to 20 GB of RAM. The final graph has 9.6 million nodes and weights 5.4 GB on disk. Building the *C elegans* pangenome took 10 minutes wallclock and uses 385 MB on disk. The query response times on these graphs ranged between 0 and 6 seconds.

Discussion

Using human RNA-seq data, we demonstrate Vizitig's capability to uncover different variants in an alignment-free, unified framework. Vizitig efficiently integrates raw sequencing data and annotated references, and promotes simultaneous manual inspection of sequence datasets.
Future developments should focus on improving core data structure scalability through integration with advanced, high-performance database engines. Implementing graph simplification strategies, such as graph coarsening, will enable clearer multi-layer visualizations, and improve large scale, pangenomics visualization - hence allowing us to handle `gfa` graphs as inputs. Finally, we would like to automatize pattern discovery processes to translate frequently observed graph patterns into ready-to-use queries could substantially improve user accessibility.

Availability and implementation

Vizitig is available as an open-source software, and documentation can be found at: `https://gitlab.inria.fr/vizisoft/vizitig`.

Funding information

This work is funded by ANR Find-RNA ANR-23-CE45-0003.

Acknowledgements

The authors would like to thank Margaux Mouton, Loup Lobet and Loïc Marmey who helped testing Vizitig v0.$*$ to 1.0, and Bruno Guillon for his help regarding NetworkDisk.

Figures

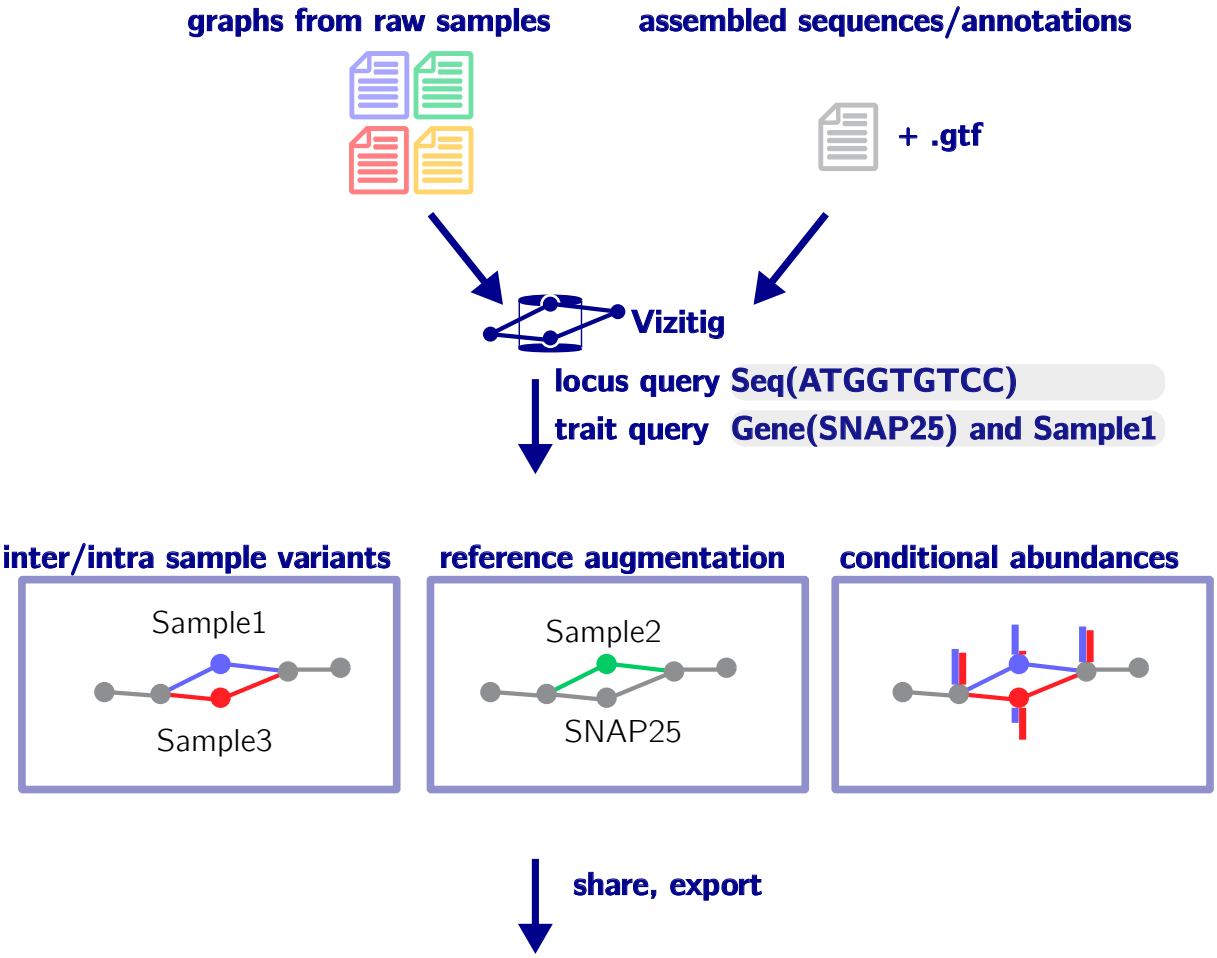

Fig. 1. *An overview of Vizitig.*

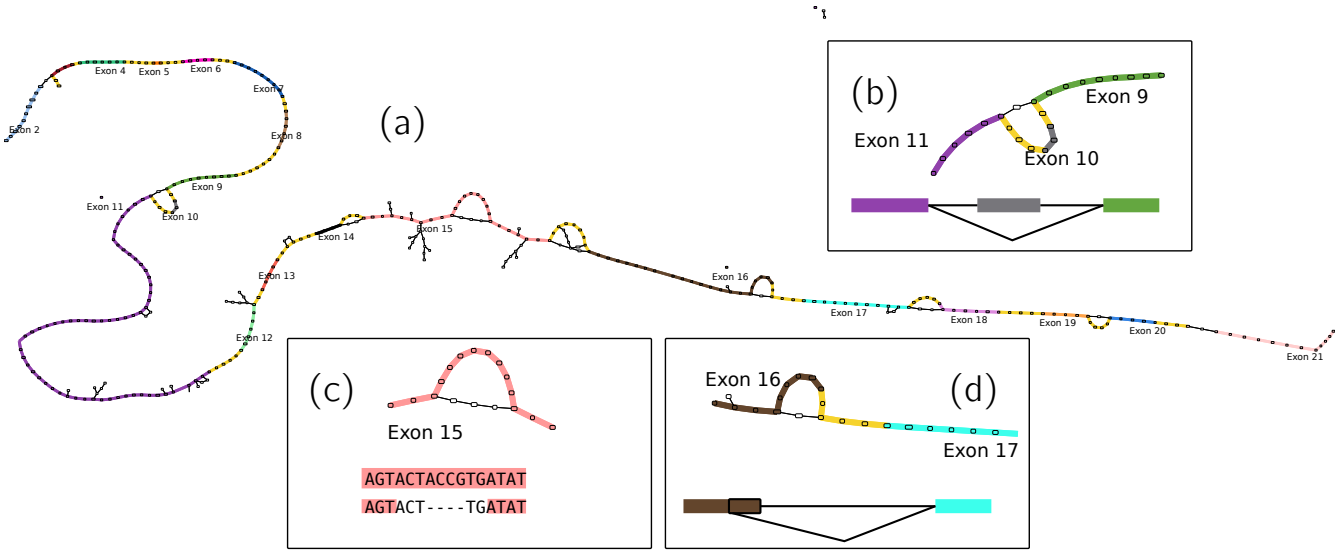

Fig. 2. *(a) A CIC transcript expressed sequence in 3 samples, built from raw RNA-seq reads and annotated with GRCh38. Exons are in different colors and labeled by number, exon-exon junctions are in yellow. This figure is directly extracted from Vizitig by exporting a SVG file. (b) Example of putative of alternative exon splicing of exon 10. A novel, uncolored path makes a direct junction between exons 11 and 9. (c) Putative novel indel in exon 15. (d) Alternative splicing start/end start in exon 16. The regular end of exon 16/junction to exon 17 is bypassed and starts earlier in exon 16.*

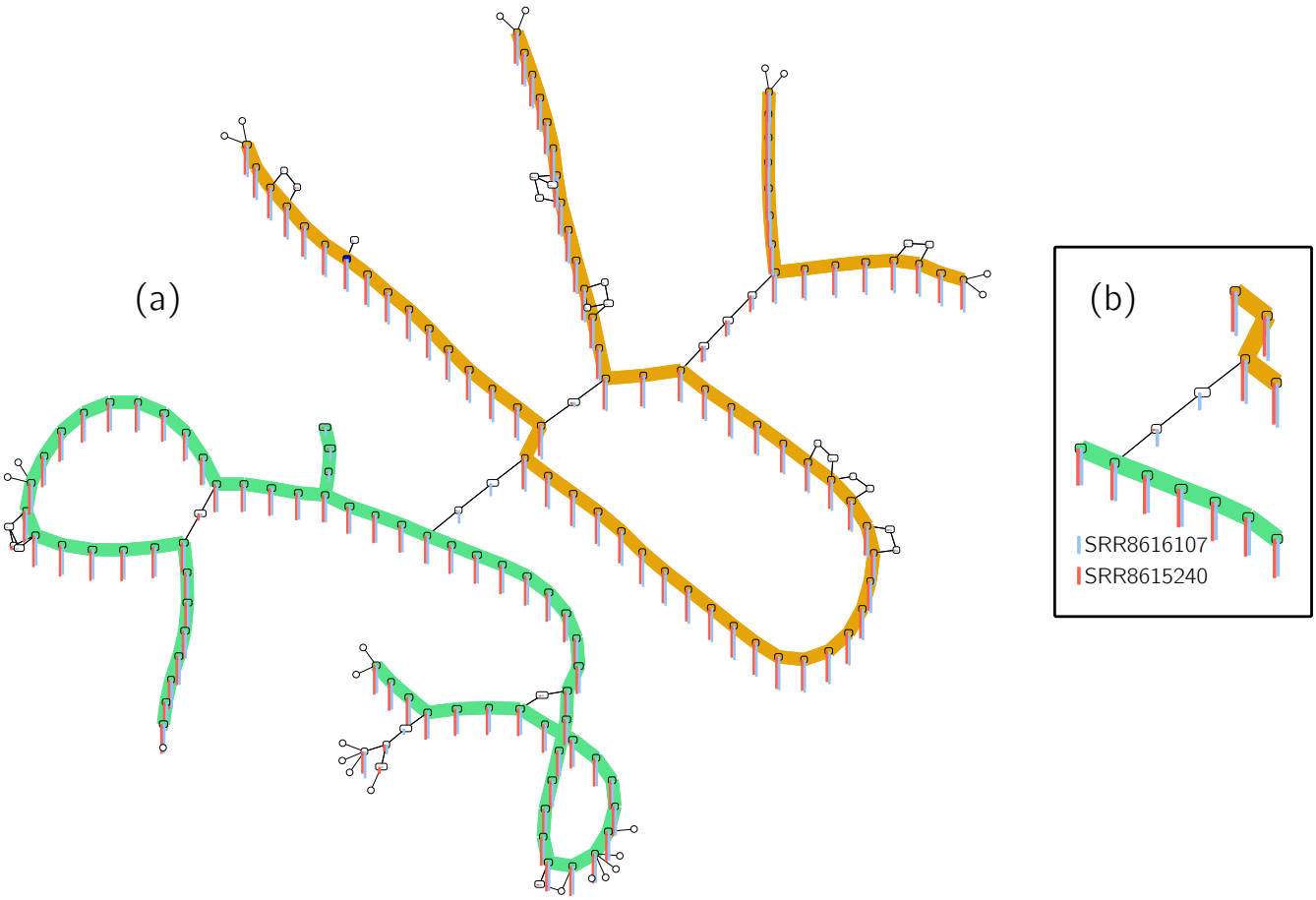

Fig. 3. *(a) The fusion region after a locus-based query and graph expand. The two genes are in yellow and green. Blue and red bars give abundance estimates in respectively SRR8615240 and SRR8616107. (b) Zoomed-in fusion event, with nodes belonging to node annotation (uncolored) expressed in SRR8616107.*

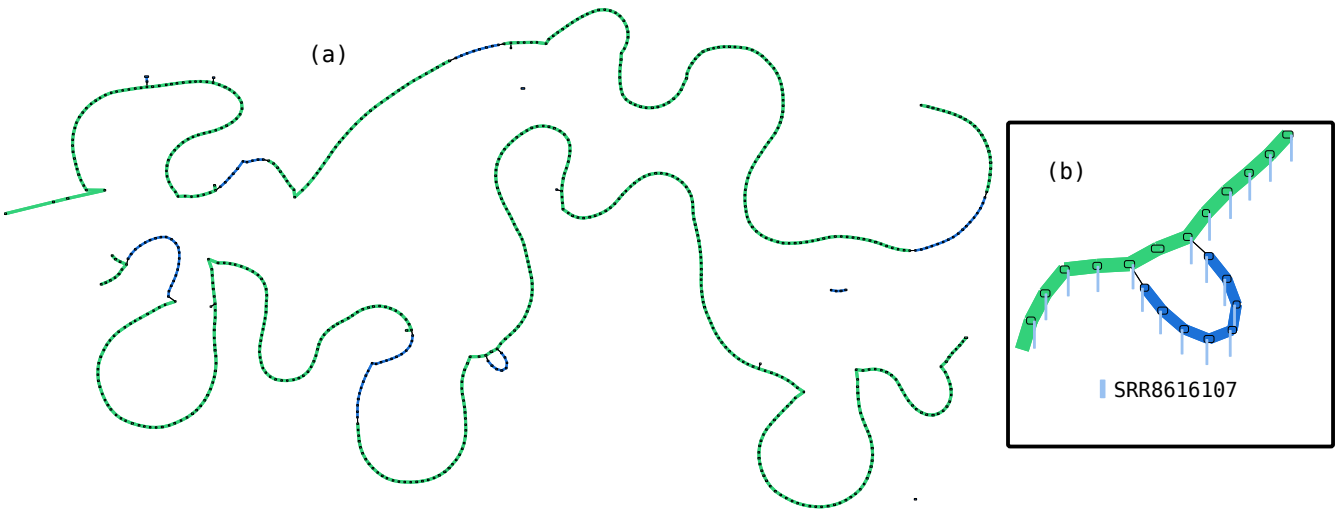

Fig. 4. *(a) Expressed sequence of gene AP3D1 after a trait-based query to find regions where sample SRR8616107 is uniquely and highly expressed. The two long paths are annotated with the same transcript names (e.g.* NM_00393 *which means that the sequence could not be assembled in one part with our chosen k value). Novel sequences absent from the references are in blue, annotated regions of AP3D1 in green (b) Zoomed-in indel event, with nodes expressed only in SRR8616107 in blue. Abundance showed in bars reveals that the wild type (green) is never expressed in SRR8616107.*

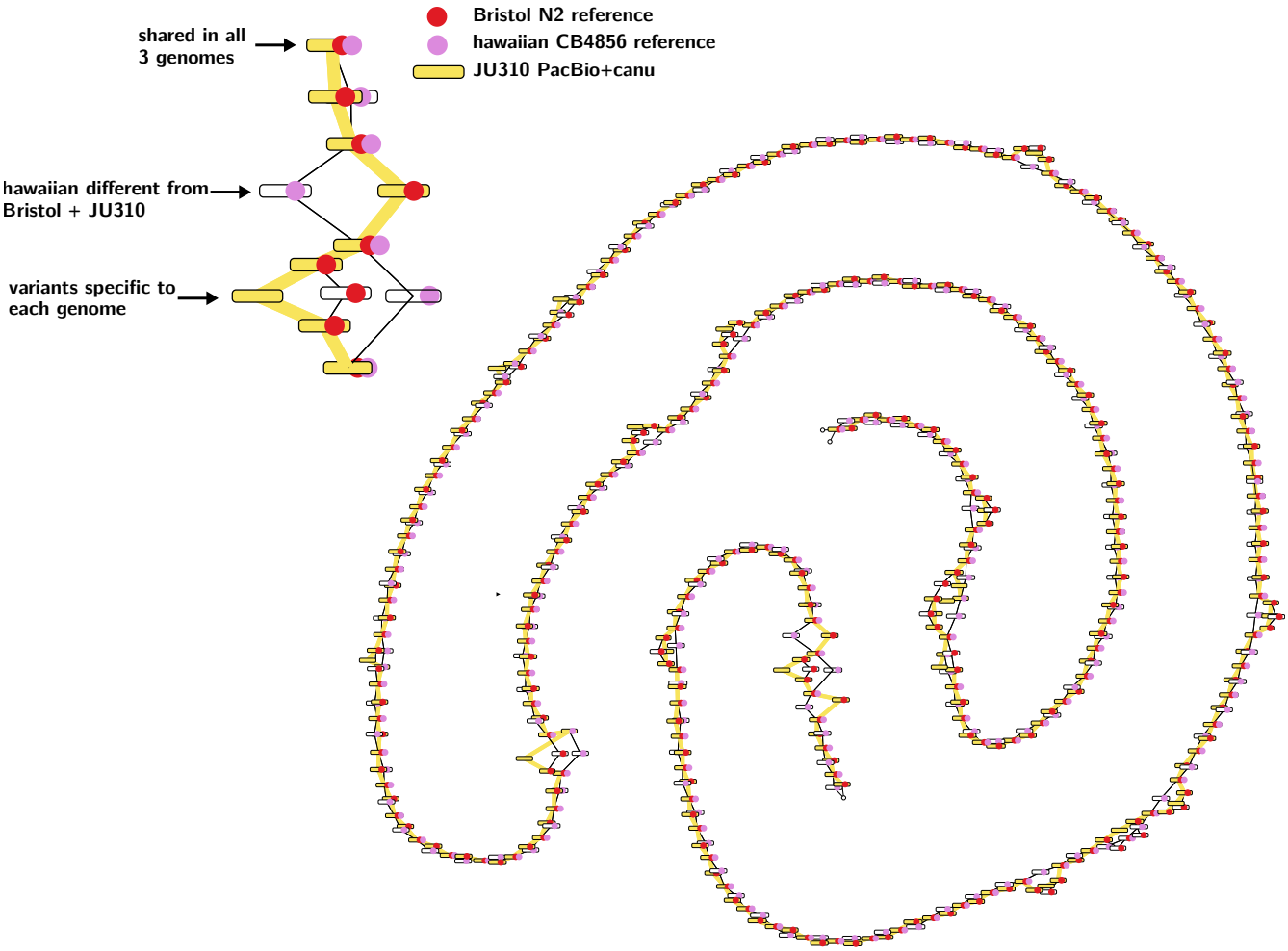

Fig. 5. *Extract of ∼ 10% of a C elegans pangenome made of two strains, augmented with PacBio long reads for a third strain JU310. We zoom in an area that features different combinations of small structural variants (indels) between the three strains in the gla-3 dhs-3 genomic region of Chr I.*

References

[1] Chikhi R, Raffestin B, Korobeynikov A, Edgar R, Babaian A. Logan: planetary-scale genome assembly surveys life's diversity. bioRxiv. 2024:2024-07.

[2] Karasikov M, Mustafa H, Danciu D, Zimmermann M, Barber C, Rätsch G, et al. Indexing All Life's Known Biological Sequences. bioRxiv. 2024.

[3] Marchet C, Boucher C, Puglisi SJ, Medvedev P, Salson M, Chikhi R. Data structures based on k-mers for querying large collections of sequencing data sets. Genome research. 2021;31(1):1-12.

[4] Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, Kulikov AS, et al. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. Journal of computational biology. 2012;19(5):455-77.

[5] Andreace F, Lechat P, Dufresne Y, Chikhi R. Comparing methods for constructing and representing human pangenome graphs. Genome Biology. 2023;24(1):274.

[6] Iqbal Z, Turner I, McVean G. High-throughput microbial population genomics using the Cortex variation assembler. Bioinformatics. 2013;29(2):275-6.

[7] Sacomoto GA, Kielbassa J, Chikhi R, Uricaru R, Antoniou P, Sagot MF, et al. K is s plice: de-novo calling alternative splicing events from rna-seq data. In: BMC bioinformatics. vol. 13. Springer; 2012. p. 1-12.

[8] Horsfield ST, Tonkin-Hill G, Croucher NJ, Lees JA. Accurate and fast graph-based pangenome annotation and clustering with ggCaller. Genome Research. 2023;33(9):1622-37.

[9] Bessière C, Zamani A, Pfeifer R, Dailhau S, Marchet C, Guibert B, et al. A strong internal promoter drives massive expression of YEATS-domain devoid MLLT3 transcripts in HSC and most lethal AML. Cancer communications (London, England).

[10] Wick RR, Schultz MB, Zobel J, Holt KE. Bandage: interactive visualization of de novo genome assemblies. Bioinformatics. 2015;31(20):3350-2.

[11] Durant É, Sabot F, Conte M, Rouard M. Panache: a web browser-based viewer for linearized pangenomes. Bioinformatics. 2021;37(23):4556-8.

[12] Jonkheer EM, van Workum DJM, Sheikhizadeh Anari S, Brankovics B, de Haan JR, Berke L, et al. PanTools v3: functional annotation, classification and phylogenomics. Bioinformatics. 2022;38(18):4403-5.

[13] Guarracino A, Heumos S, Nahnsen S, Prins P, Garrison E. ODGI: understanding pangenome graphs. Bioinformatics. 2022;38(13):3319-26.

[14] Chikhi R, Limasset A, Medvedev P. Compacting de Bruijn graphs from sequencing data quickly and in low memory. Bioinformatics. 2016;32(12):i201-8.

[15] Consortium CCLE, of Drug Sensitivity in Cancer Consortium G, et al. Pharmacogenomic agreement between two cancer cell line data sets. Nature. 2015;528(7580):84-7.

[16] Gillani R, Seong BKA, Crowdis J, Conway JR, Dharia NV, Alimohamed S, et al. Gene fusions create partner and collateral dependencies essential to cancer cell survival. Cancer research. 2021;81(15):3971-84.

[17] Kent WJ. BLAT—the BLAST-like alignment tool. Genome research. 2002;12(4):656-64.

[18] Lee D, Zdraljevic S, Stevens L, Wang Y, Tanny RE, Crombie TA, et al. Balancing selection maintains hyper-divergent haplotypes in Caenorhabditis elegans. Nature ecology & evolution. 2021;5(6):794-807.