Telomere-to-telomere assembly by preserving contained reads

Sudhanva Shyam Kamath, Mehak Bindra, Debnath Pal, et al.

Genome Res. published online October 15, 2024
Access the most recent version at doi:10.1101/gr.279311.124

Telomere-to-telomere assembly by preserving contained reads

Sudhanva Shyam Kamath, Mehak Bindra, Debnath Pal, and Chirag Jain

Department of Computational and Data Sciences

Indian Institute of Science, Bangalore 560012, India

Abstract. Automated telomere-to-telomere (T2T) *de novo* assembly of diploid and polyploid genomes remains a formidable task. A string graph is a commonly used assembly graph representation in the assembly algorithms. The string graph formulation employs graph simplification heuristics, which drastically reduce the count of vertices and edges. One of these heuristics involves removing the reads contained in longer reads. In practice, this heuristic occasionally introduces gaps in the assembly by removing all reads that cover one or more genome intervals. The factors contributing to such gaps remain poorly understood. In this work, we mathematically derived the frequency of observing a gap near a germline and a somatic heterozygous variant locus. Our analysis shows that (i) an assembly gap due to contained read deletion is an order of magnitude more frequent in Oxford Nanopore reads than PacBio HiFi reads due to differences in their read-length distributions, and (ii) this frequency decreases with an increase in the sequencing depth. Drawing cues from these observations, we addressed the weakness of the string graph formulation by developing the RAFT assembly algorithm. RAFT addresses the issue of contained reads by fragmenting reads and producing a more uniform read-length distribution. The algorithm retains spanned repeats in the reads during the fragmentation. We empirically demonstrate that RAFT significantly reduces the number of gaps using simulated datasets. Using real Oxford Nanopore and PacBio HiFi datasets of the HG002 human genome, we achieved a twofold increase in the contig NG50 and the number of haplotype-resolved T2T contigs compared to Hifiasm.

Introduction

Building high-quality haplotype-resolved *de novo* assemblies remains a principal challenge in genomics research. The T2T (telomere-to-telomere) assembly of CHM13 human genome (Nurk et al., 2022) is a recent scientific milestone which has inspired further efforts toward achieving T2T assembly of personal diploid human genomes (Jarvis et al., 2022; Yang et al., 2023). Third-generation read sequencing technologies like Pacific Biosciences high-fidelity (PacBio HiFi) reads and Oxford Nanopore Technology (ONT) reads were instrumental in constructing the CHM13 reference genome. Currently, PacBio HiFi and ONT Duplex se-

quencing technologies produce reads that have an average length greater than 10 kbp and per-base error rates less than 0.5% (Li and Durbin, 2024; Wenger et al., 2019).

De novo genome assembly using long reads is most commonly solved using overlap-layout-consensus based methods (Myers, 1995). The assembly workflow typically involves (i) computing pairwise overlaps between reads, (ii) error-correction of reads, (iii) constructing a read-overlap graph, and (iv) identifying walks in the graph which correspond to contiguous substrings of the genome. In a read-overlap graph, the reads are represented as vertices, and the suffix-prefix overlaps between the reads are represented as directed edges. The initial version of this graph is quite tangled and requires additional graph simplification heuristics to remove redundant vertices and edges. Myers's *string graph* formulation (Myers, 2005, 1995) has long been the standard choice to build a simplified version of a read-overlap graph (Baaijens et al., 2017; Cheng et al., 2024, 2021, 2022; Chin et al., 2013, 2016; Feng et al., 2022; Koren et al., 2017; Li, 2016; Vaser and Šikić, 2021; Vicedomini et al., 2021). The string graph model was also used by the T2T Consortium to assemble the CHM13 human genome (Nurk et al., 2022).

The two important graph-simplification steps in the string graph formulation are (i) removal of transitively inferable edges and (ii) deleting those reads that are entirely contained as a substring in another read (Myers, 2005). The advantage of these two steps is that they prohibit redundancy, i.e., no two walks in a string graph spell the same sequence (Tomescu and Medvedev, 2017). These are crucial simplification steps that simplify the topology of the overlap graph and reduce the computational cost of identifying long unambiguous contigs from the overlap graph.

However, prior works have highlighted that the removal of contained reads from the graph is an 'unsafe' operation because this heuristic can occasionally disconnect the walks corresponding to true chromosome sequences (Cheng et al., 2024; Feng et al., 2022; Hui et al., 2016; Jain, 2023; Li and Durbin, 2024; Nurk et al., 2022). The connectivity breaks when all reads that cover one or more genomic intervals are removed (Figure 1A). We refer to these events as assembly gaps due to contained read deletion (formally defined in Methods). The need to remove contained reads is currently a major weakness of the read-overlap-based assembly algorithms (Li and Durbin, 2024).

A few approaches have been proposed to tackle the above problem. The algorithms by Hui et al. (2016) and Jain (2023) work under simplified assumptions on input read lengths and sequencing coverage. These algorithms don't trivially extend to practical solutions for assembling highly repetitive genomes. An initial release of the Hifiasm assembler (Cheng et al., 2021) included a method to recover an essential contained read if the read connects the ends of two walks in the graph. This technique has been observed to work in simple

scenarios but is not always reliable (Jain, 2023; Li and Durbin, 2024). A more recent version of Hifiasm also uses alignments of ultra-long nanopore reads to a string graph to identify the necessary contained reads (Cheng et al., 2024). This is a useful approach if ultra-long reads are available. Previous experiments (Jain, 2023) have reported that contained read deletion in a string graph is more likely to impact graph connectivity in the regions of low heterozygosity. In such regions, a longer read sampled from one haplotype may contain all reads that cover the homologous region in the opposite haplotype.

A sequencing run results in a multiset of reads. Considering all possible distinct sequencing outputs, we mathematically derived a formula to calculate the fraction of sequencing outputs in which an assembly gap would occur due to contained read deletion. It is useful to compare the fractions in different experimental settings, e.g., with different choices of sequencing technology and sequencing depth. We performed this theoretical analysis for both normal and cancer genomes. The analysis reveals novel insights into the key factors contributing to assembly gaps due to contained read deletion. We refer to this method as CGProb (`https://github.com/at-cg/CGProb`).

Next, using insights from CGProb, we developed RAFT (Repeat-Aware Fragmenting Tool, `https://github.com/at-cg/RAFT`) to prevent assembly gaps during genome assembly. Conceivably, the proportion of contained reads in a sequencing dataset is roughly determined by its read-length distribution. On the one hand, there are no contained reads if all reads have a fixed length, whereas an ONT sequencing dataset may have a significant fraction of contained reads due to a wide read-length distribution (Logsdon et al., 2020). RAFT reduces the range of read lengths in a sequencing dataset by fragmenting long reads into equal-sized shorter reads. The reads predicted to span repetitive regions of the genome are treated differently. RAFT enables high-quality phased assemblies of variable-length long and accurate reads (e.g., ONT Duplex reads or a mixture of ONT Duplex and PacBio HiFi reads). Both of the above tools, RAFT and CGProb, are useful in the era of telomere-to-telomere genomics.

Results

Overview of CGProb

Haplotype-resolved assembly of diploid genomes is challenging because one needs to distinguish between reads originating from two near-identical haplotype sequences. The differences between the haplotypes occur at heterozygous loci. Contained read deletion may break haplotype walks in a read-overlap graph. We show an example in Figure 1A where an assembly gap occurs in the second haplotype after the deletion of contained

read r_8. Assembly gaps can occur in haploid genomes due to repeats as well (Supplemental Figure S1). For brevity, we refer to the assembly gaps due to contained read deletion as just 'assembly gaps' in this section. The occurrence of assembly gaps in a string graph depends on several factors, including the sampling positions of reads, genome heterozygosity, sequencing coverage, ploidy, repetitive sequences, etc. As a result, deriving the expected number of assembly gaps in a string graph is challenging. Knowing this value for different choices of sequencing technology and sequencing depth can allow a more informed decision-making for *de novo* genome sequencing.

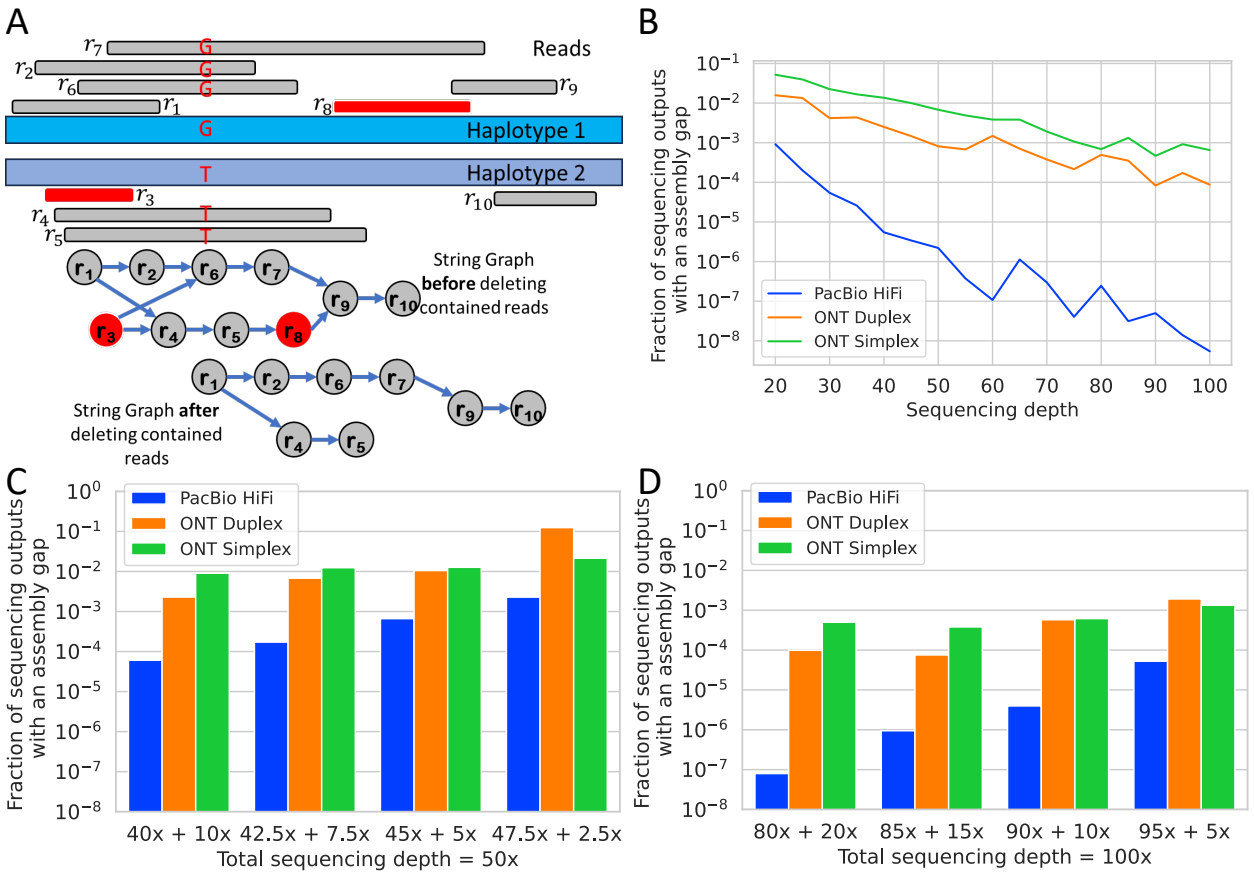

Fig. 1: Assembly gaps and their occurrence frequency. (A) An example of a sequencing output where an assembly gap occurs in the string graph due to contained read deletion. Read r_3 is contained in read r_1. Read r_8 is contained in read r_7. Accordingly, the string graph representation excludes reads r_3 and r_8. Read r_3 is redundant; its deletion simplifies the graph. However, removing read r_8 breaks the connectivity between reads r_5 and r_9, which was necessary to spell the second haplotype. **(B)** Fraction of sequencing outputs containing an assembly gap. We measured the fractions using the read-length distributions corresponding to three sequencing technologies (PacBio HiFi, ONT Duplex, ONT Simplex) and using different sequencing depths. Here we used equal sequencing depths on both haplotypes. **(C)** and **(D)** Fraction of sequencing outputs containing an assembly gap when the sequencing depths across the two haplotypes are uneven. This scenario models somatic mutation in DNA with variant allele frequency below 0.5. In panel (C), the total sequencing depth for both haplotypes is 50×. In panel (D), the total sequencing depth is 100×.

Consider the output of a sequencing experiment as a multiset of reads, where each read is identified by its haplotype of origin, length, and stop position. The user provides a read-length distribution and sequencing depths of two haplotypes as input. Accordingly, the set of valid sequencing outputs includes all possible multisets of read-sampling intervals that are consistent with the user-provided input (Methods). CGProb considers all these valid sequencing outputs and calculates the fraction of sequencing outputs in which an assembly gap occurs (Methods). We made a few simplifying assumptions to make the analysis feasible: e.g., (i) there is a single heterozygous SNP locus in the diploid genome, (ii) reads are error-free, and (iii) the two haplotypes do not have repetitive DNA (Methods). Although these assumptions will likely not hold in practice, the above model is informative to study the frequency of an assembly gap near an isolated heterozygous locus while assembling error-corrected long reads. A naive method to calculate the fraction would check all possible $O(G^N)$ read sequencing outputs individually, where G is the genome length and N is the number of reads. Instead, we developed an efficient combinatorial technique to count the sequencing outputs containing an assembly gap in polynomial time. The theory and implementation details are provided in the Methods. We further validated the accuracy of CGProb using simulation experiments (Supplemental Table S1).

Frequency of observing an assembly gap

Using CGProb, we evaluated the frequency of observing an assembly gap near a heterozygous SNP. In the first scenario, the heterozygous SNP is a germline mutation. Here we used equal sequencing depths for both haplotypes (paternal and maternal). In the second scenario, evidence for a heterozygous SNP is observed in the sequencing output due to a somatic mutation with variant allele frequency below 0.5. Variant allele frequency is the fraction of reads supporting a specific DNA variant divided by the overall coverage at that locus. Here, we used uneven sequencing depths for the two haplotypes (e.g., tumour and normal). We considered three prominent sequencing technologies: PacBio HiFi, ONT Simplex, and ONT Duplex. For each sequencing depth and for each technology, we simulated multiple read length distributions consistent with that sequencing technology and depth of coverage (Methods). We ran CGProb on each read length distribution and recorded the median fraction. The minimum and maximum values are shown separately in Supplemental Figures S2, S3.

(1) Germline heterozygous variant locus: We computed the fraction of the sequencing outputs containing an assembly gap while varying the genome sequencing depths from 20× to 100× (Figure 1B). Here the sequencing depths were balanced equally on both haplotypes, e.g., 20× depth corresponds to 10× depth on

each haplotype. Our results show that there is at least an order of magnitude difference in the fraction of sequencing outputs containing an assembly gap for ONT reads compared to PacBio HiFi reads. The results imply that assembly gaps are more frequent when there is a larger variation in read lengths. The read-length distribution of ONT reads is generally more skewed than PacBio HiFi reads (Supplemental Figure S4). Intuitively, the fraction of contained reads will be greater if the variation in read lengths is larger.

Figure 1B also shows a decrease in the median fraction as sequencing depth increases, although some deviation from this trend is observed for PacBio HiFi reads due to noise arising from our use of a small number of trials. This decreasing trend is also intuitive because if the sequencing coverage is higher, then the number of times a genome interval is sequenced becomes larger which reduces the chance of every read which supports that interval being a contained read.

(2) Somatic heterozygous variant locus: We analysed the fraction of the sequencing outputs containing an assembly gap in a simulated heterogeneous sequencing sample, e.g., a sample with mixed normal and tumour cells. We set total sequencing depths as $50\times$ and $100\times$. We set the tumour sequencing depths as $5\%, 10\%, 15\%$, and 20% of the total (Figures 1C, 1D). For all three sequencing technologies, we observed that the fraction of sequencing outputs containing an assembly gap increases as the tumour sequencing depth decreases. The result implies that assembly gaps are more frequent near somatic genetic variants with lower variant allele frequencies. This is expected because all the reads sampled from an interval are more likely to be contained in a read from the second homologous interval if the coverage over the first interval is low and the coverage over the second interval is high. We again found that a string graph of ONT reads is more likely to contain an assembly gap than a string graph of PacBio HiFi reads.

Overview of RAFT

The above analysis indicates that the problem of assembly gaps due to contained read deletion is much less prevalent with narrow read-length distributions. Inspired by these results, we developed RAFT as a practical solution to assemble a long-read dataset when there is significant variability in the read lengths. The RAFT algorithm fragments long reads into shorter, uniform-length reads while also considering the potential usefulness of the longer reads in assembling complex repeats. We envision RAFT as a module that can be easily integrated into any existing overlap-layout-consensus-based assembler.

The input to the RAFT algorithm includes error-corrected long reads and all-to-all pairwise alignment information (Figure 2A). The algorithm carefully fragments the input reads. While fragmenting a read r, we consider its high-quality pairwise alignments with other reads. If the number of alignments to an interval in

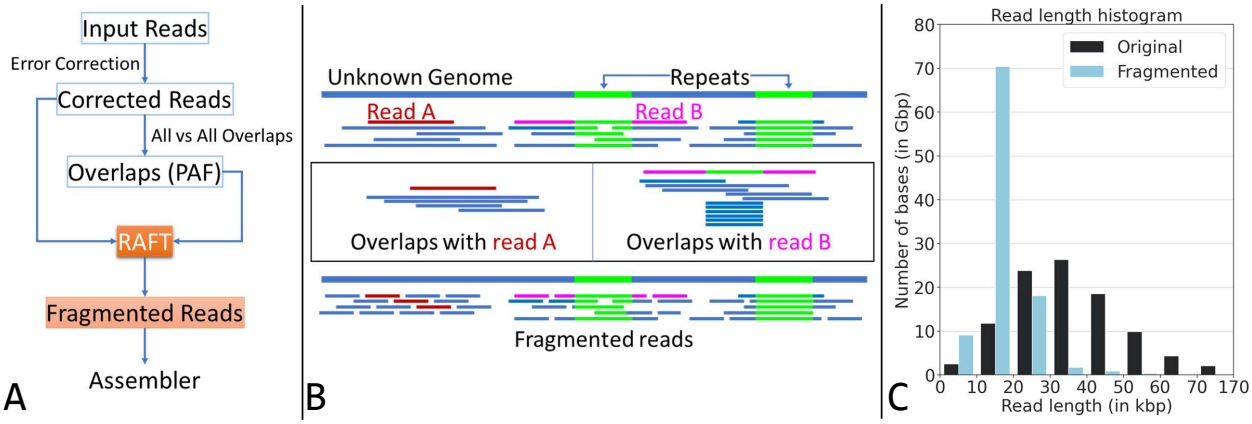

Fig. 2: Illustration of the RAFT algorithm and its usage for genome assembly. (A) Flowchart of an assembly workflow that uses RAFT. RAFT accepts error-corrected long reads and all-to-all alignment information as input. It produces a revised set of fragmented reads with a narrow read-length distribution. **(B)** Illustration of the RAFT algorithm. Read A (shown in red) is sampled from a non-repetitive region of the genome. Accordingly, RAFT fragments read A into shorter uniform-length reads. Read B (shown in pink) spans a repetitive region of the genome. RAFT detects the repetitive interval in read B because more than the expected number of sequences align to that interval. The portions of read B outside the repetitive interval are split into shorter reads. **(C)** The impact of RAFT can be seen on a set of ONT Duplex reads sampled from the HG002 human genome. The range of the read-lengths is significantly reduced by using RAFT. The original dataset comprises 3.7 million reads with a skewed read length distribution. After fragmentation, the dataset comprises 6.8 million reads.

read r exceeds $\mu \cdot cov$, where μ is a user-specified threshold (default value = 1.5) and cov is the coverage of the input sequencing dataset, then we prevent the interval from being fragmented. Such intervals are potentially repetitive and may be necessary to resolve repetitive sequences (Figure 2B). We set the default length of fragmented reads as 20 kbp to ensure that the fragmented read lengths are greater than the lengths of the abundant interspersed repeats such as LINEs (Methods).

RAFT fits conveniently in a *de novo* genome assembly workflow in between the long-read error correction step and the assembly steps (Figure 2A). To evaluate RAFT, we designed the 'RAFT-Hifiasm' workflow that combines RAFT's ability to manipulate the read-length distribution and Hifiasm's highly efficient all-to-all alignment and error-correction algorithms (Cheng et al., 2021). Accordingly, the RAFT-Hifiasm workflow uses Hifiasm for error-correction of input reads and computing all-vs-all pairwise read alignments. RAFT uses this information to generate a set of fragmented reads (Figure 2C). In the end, we assemble the fragmented reads using Hifiasm.

Evaluation using simulated data

We simulated error-free long reads from a publicly available haplotype-resolved HG002 diploid human genome assembly using Seqrequester (Walenz, 2023). We simulated one PacBio HiFi (30×), two ONT Simplex (30×,

50×), and two ONT Duplex (30×, 50×) datasets. The read-length distributions of these sequencing datasets are consistent with real long-read sequencing data (Supplemental Table S2). We consider a shorter read as contained in a longer read if the shorter read is a proper substring of the longer read. A read is *non-contained* if it is not contained in any other read.

		PacBio HiFi	ONT Simplex		ONT Duplex	
		30×	30×	50×	30×	50×
Assembly gaps due to contained read deletion	Standard string graph	5	51	11	30	10
	Hifiasm	3	40	6	17	4
	RAFT-Hifiasm	0	1	0	2	1
Fraction of bases used in string graph	Standard string graph	44.54%	23.87%	17.94%	30.91%	22.93 %
	Hifiasm	44.55%	24.09%	18.02%	30.92%	22.94 %
	RAFT-Hifiasm	73.27%	61.01%	57.22%	65.88%	62.04 %

Table 1: Evaluation of RAFT-HiFiasm using simulated data. We show the count of assembly gaps in the string graphs constructed by three methods, and the fraction of bases used in the graphs. The standard string graph is based on the original string graph formulation (Myers, 2005) that ignores contained reads. The row labelled Hifiasm is the string graph which contains contained reads that were rescued by a heuristic implemented in Hifiasm. The row RAFT-Hifiasm considers the string graph constructed after fragmenting reads using RAFT and then assembling the fragmented reads using Hifiasm.

We tested RAFT-Hifiasm and Hifiasm methods to evaluate their ability to address the issue of assembly gaps that occur due to contained read deletion. The standard string graph formulation (Myers, 2005) uses non-contained reads and ignores contained reads. Hifiasm (Cheng et al., 2021) uses non-contained reads to build its initial string graph and rescues a small number of contained reads later. In the RAFT-Hifiasm method, RAFT outputs a set of fragmented reads. The string graph is constructed using non-contained reads in the fragmented sequencing data. Again, Hifiasm attempts to rescue some contained reads. The benefit of using simulated data in this experiment is that we know the sampling interval of the reads in the original genome sequence.

One way to spot an assembly gap due to contained read deletion is by aligning the set of reads retained in a string graph to the HG002 genome. Any interval in the genome which has zero read-alignment coverage but non-zero sequencing depth (with respect to the original set of reads) corresponds to an assembly gap caused by contained read deletion (Methods).

RAFT-Hifiasm outperformed Hifiasm in this experiment. Using RAFT-Hifiasm, we were able to reduce the number of assembly gaps after contained read deletion by at least an order of magnitude. We eliminated the gaps entirely in two datasets (Table 1). A small number of assembly gaps due to contained read deletion

remain when using RAFT-Hifiasm because RAFT preserves repetitive regions in reads. RAFT increases the fraction of non-contained bases by narrowing the read-length read distribution. Accordingly, the fractions of bases used in RAFT-Hifiasm's graphs are higher. In all methods, the unused bases in the graph were still used in the initial stages of the assembly, e.g., for read error correction, chimeric read detection, etc. We demonstrate the impact of the RAFT approach on improving assembly quality in the next section.

Evaluation using real human sequencing data

We tested the RAFT-Hifiasm workflow using four publicly available real datasets comprising long and accurate reads sampled from the HG002 human genome. The first dataset, D1, comprises PacBio HiFi reads with 36× coverage. Dataset D2 is an ONT Duplex sequencing dataset with 32× coverage. Dataset D3 is a combination of D1 and D2 datasets; thus, its coverage is 68×. Dataset D4 is a high-accuracy ultra-long ONT dataset with 40× coverage. The read-length statistics of these datasets are available in Supplemental Table S2. We assembled these datasets using three methods: (i) Hifiasm, (ii) NaiveCut-Hifiasm, and (iii) RAFT-Hifiasm to compare their output. In the NaiveCut-Hifiasm method, we fragment reads in a 'repeat-agnostic' manner, that is, we fragment the reads into the same length as RAFT while assuming there are no repeats. The commands and software versions for reproducing the analysis are listed in Supplemental Note S3. We skipped a detailed comparison with other recent long-read assemblers such as Verkko (Rautiainen et al., 2023), and LJA (Bankevich et al., 2022). Both Verkko and LJA are de Bruijn graph-based assemblers and, as such, do not share the limitations of contained read deletion caused in the string graph-based assemblers like Hifiasm.

We expected improvements in the datasets comprising ONT reads (D2-D4) because these datasets have wide read-length distributions (Supplemental Figure S4). The results obtained using the Hifiasm, NaiveCut-Hifiasm, and RAFT-Hifiasm methods are shown in Table 2. Applying the RAFT algorithm on datasets D2-D4 improved the assembly contiguity. The NG50 statistic is defined such that 50% of the estimated size of assembly (3.1 Gbp) is realized by contigs of NG50 length or longer. The RAFT-Hifiasm method generated more contiguous assemblies using datasets D2-D4 as indicated by the contig NG50 metric and the count of T2T-complete contigs. We note that NaiveCut-Hifiasm produced more contiguous assemblies than Hifiasm but failed to beat RAFT-Hifiasm. This suggests that our "repeat-aware" fragmentation strategy led to better assembly contiguity. The switch error rate was not significantly impacted by RAFT's read fragmentation, which suggests that Hifiasm does not face any additional difficulty phasing the fragmented reads during assembly. Compared to the assemblies produced by Hifiasm and for datasets D2-D4, the assemblies obtained

Dataset	Method	Size (Gbp)	NG50 (Mbp)	Switch error (%)	T2T contigs	Multicopy genes retained(%)	Gene completeness Complete (%)	Gene completeness Duplicated (%)
D1: HiFi (36×)	Hifiasm	3.04/3.04	59.0/45.4	1.09/0.96	0	83.37/76.42	97.59/97.70	0.45/0.31
	NaiveCut-Hifiasm	3.06/2.96	51.1/48.7	1.12/0.95	0	80.50/80.42	97.66/97.57	0.45/0.39
	RAFT-Hifiasm	3.04/2.98	44.9/62.4	1.06/1.01	1	79.22/82.09	97.87/97.45	0.34/0.39
D2: ONT Duplex (32×)	Hifiasm	2.99/3.04	42.2/51.0	2.37/1.66	2	80.98/81.45	96.94/96.63	1.06/1.37
	NaiveCut-Hifiasm	3.01/3.00	61.2/56.3	2.04/2.08	2	83.61/77.22	97.61/97.64	0.54/0.42
	RAFT-Hifiasm	2.96/3.04	80.3/52.6	2.02/1.90	6	83.53/80.98	97.72/97.59	0.50/0.50
D3: HiFi (36×) + ONT Duplex (32×)	Hifiasm	3.13/2.99	49.3/53.3	0.82/1.00	1	83.85/77.78	95.69/95.79	2.42/2.10
	NaiveCut-Hifiasm	3.04/3.01	81.9/82.1	1.02/1.08	2	83.61/78.74	97.93/97.76	0.40/0.42
	RAFT-Hifiasm	3.03/3.02	89.6/89.3	0.94/1.10	7	83.13/80.50	97.79/97.98	0.42/0.41
D4: ONT high-acc UL (40×)	Hifiasm	3.44/3.43	16.2/20.8	2.49/1.69	0	74.02/78.18	67.72/70.41	19.88/19.82
	NaiveCut-Hifiasm	3.02/3.11	45.2/51.7	1.96/2.05	0	81.14/79.54	97.07/96.91	0.55/0.60
	RAFT-Hifiasm	3.05/3.07	81.3/49.1	2.19/1.87	1	81.45/77.45	97.21/96.94	0.71/0.51

Table 2: Evaluation of the RAFT-Hifiasm workflow for computing haplotype-resolved assembly. We measured assembly quality statistics separately for both haplotypes. The reported statistics are formatted as haplotype 1/haplotype 2. In the NaiveCut-Hifiasm method, we fragment all reads to the same length as RAFT, regardless of whether the read contains a repetitive region. NG50 is the length of the shortest contig at 50% of the genome length. We assumed a genome length of 3.1 Gbp. Switch error is the percentage of incorrectly phased adjacent SNP sites. A contig is defined as T2T if it contains the telomeric repeat unit "TTAGGG" within 1kbp of both ends, and aligns with a reference chromosome with more than 95% identity. 'Multicopy genes retained' is the percentage of multicopy genes in GRCh38, i.e. genes with multiple mapping positions at $\geq$ 99% sequence identity, that occur multiple times in the assembly. In the gene completeness statistics, the percentage of complete genes are those genes occurring only once in the assembly only once in GRCh38 (at 99% sequence identity). The percentage of duplicated genes are those genes which occur multiple times in the assembly and occur only once in GRCh38. The tools and commands used to measure the assembly statistics are available in Supplemental Note S3.

by RAFT-Hifiasm improved gene completeness and reduced the percentage of false duplications. Further evaluation of the assemblies, including an assessment using the Genome in a Bottle benchmark (Zook et al., 2020), is provided in Supplemental Table S3.

We also assembled datasets D1, D2, and D3 with complementary parental Illumina sequencing data to obtain extended phasing. We evaluated Hifiasm, RAFT-Hifiasm, and Verkko assemblers. In this experiment, Verkko produced the most contiguous assemblies but it also resulted in a slightly higher rate of false duplications in single-copy genes. Summary statistics of these assemblies are described in Supplemental Table S4.

Hifiasm used 18.2 hours to assemble the largest dataset, D3, on a server with two 24-core Intel Xeon Gold 6248R CPUs (Table 3). In contrast, the RAFT-Hifiasm workflow took 38.8 hours. Currently, the RAFT-Hifiasm workflow executes RAFT once and Hifiasm thrice. The three Hifiasm runs are used for long-read error-correction, computing all-to-all read alignments, and computing genome assembly, respectively. RAFT's runtime share was about 4.8 hours. In future, tighter software integration of RAFT and Hifiasm may help avoid redundant steps and optimize runtime.

Dataset	**Hifiasm** Time / Memory (hours / GB)	**RAFT-Hifiasm workflow** Time / Memory (hours / GB)			
	Assembly	**Error correction**	**Overlap computation**	**RAFT**	**Assembly**
D1: HiFi (36x)	7.3 / 125.9	7.3 / 125.9	2.3 / 92.5	4.5 / 115.4	3.1 / 108.2
D2: Duplex (32x)	5.8 / 107.1	5.8 / 107.1	2.2 / 75.7	4.8 / 107.5	4.7 / 107.5
D3: D1+D2	18.2 / 274.6	18.2 / 274.6	6.3 / 217.4	4.8 / 291.6	9.5 / 240.7
D4: Ultralong (40x)	12.3 / 221.6	12.3 / 221.6	3.7 / 150.7	1.5 / 133.1	5.3 / 138.7

Table 3: Time and memory consumption of the Hifiasm and RAFT-Hifiasm methods. The RAFT-Hifiasm workflow involves computing the error-corrected reads, all-versus-all overlaps, fragmentation of reads, and assembly.

Evaluation using plant genome sequencing data

We assembled ONT Duplex sequencing data from two plant genomes where repetitive regions form a larger fraction of the entire genome compared to humans: (1) *Solanum lycopersicum* Heinz 1706 (tomato) and (2) *Zea mays* B73 (maize). Sequencing data for tomato genome comprised ONT Duplex reads at 37× coverage, and sequencing data for maize genome comprised ONT Duplex reads at 41× coverage (Koren et al., 2024).

We measured the NG50 and QV statistics for all assemblies. The QV value is the Phred-scaled contig base error rate measured by comparing 37-mers in the assembly contigs to 37-mers in PacBio HiFi reads from the same sample.

For the tomato genome, we observed an increase in the NG50 metric from 15.9 Mbp for Hifiasm assembly to 20.5 Mbp for RAFT-Hifiasm assembly. The assembly QV also increased from 45.9 to 46.3. Both methods, Hifiasm and RAFT-Hifiasm, resulted in comparable primary assembly sizes of 920 Mbp and 923.4 Mbp, respectively. For the maize genome, NG50 increased from 30.9 Mbp for the Hifiasm assembly to 164.4 Mbp for the RAFT-Hifiasm assembly. The assembly QV also increased from 54.6 to 58.2. Both methods, Hifiasm and RAFT-Hifiasm, resulted in comparable primary assembly sizes of 2267.3 Mbp and 2226.8 respectively. This demonstrates that using RAFT-Hifiasm results in increased assembly contiguity, even in genomes where the repeat content is high.

Discussion

This paper analyses and addresses a longstanding weakness of the string graph formulation (Myers, 2005, 1995). A string graph is a subgraph of an overlap graph constructed by removing all transitive edges and vertices corresponding to contained reads from the overlap graph. String graphs have been commonly used in several *de novo* genome and metagenome assemblers during the past three decades, but the issue of assembly gaps caused by contained read deletion came to limelight only recently (Cheng et al., 2024; Jain, 2023; Li and Durbin, 2024; Nurk et al., 2022). The quality of modern haplotype-resolved genome assemblies has improved to an extent where the few assembly gaps caused by contained read deletion are now noticeable (Li and Durbin, 2024). Contained read deletion occasionally leads to the loss of useful reads in regions of low heterozygosity of diploid or polyploid genomes and in the highly repetitive regions of genomes. In both cases, all reads that support an interval in the genome may be discarded when all of them are contained in a longer read sampled from a near-identical but different region of the genome.

We presented the CGProb method to count the frequency of observing assembly gaps due to contained read deletion in a string graph. We measured the frequency for different read-length distributions and sequencing depths. This is the first mathematical model developed to assess this problem. Our analysis showed that assembly gaps due to contained read deletion are at least one order of magnitude more frequent in ONT sequencing outputs than PacBio HiFi sequencing outputs because the latter have much less variability in read lengths. In both cases, the frequency dropped rapidly with an increase in the sequencing coverage. Our method can help users to compare the relative frequencies of an assembly gap for different sequencing

technologies at the same sequencing depth, or the same sequencing technology at different sequencing depths. CGProb currently works under the assumption of error-free reads. Accordingly, our findings are applicable to only highly accurate or well-corrected reads. We also assumed a single heterozygous locus in the diploid genome. In the subsequent versions of CGProb, we hope to further extend the theory and relax these assumptions, e.g., to compute the frequency of assembly gaps for genome sequences that are heterozygous at more than one closely-spaced loci, or for genome sequences containing repetitive sequences, or when ploidy exceeds two. Further analysis may help to characterise the regions of a genome where assembly gaps are more likely and motivate novel methods to address the issue.

We also presented RAFT as a solution to address the issue of contained reads by fragmenting reads and obtaining a more uniform read-length distribution. Using ONT Duplex reads and a mixture of ONT Duplex and PacBio HiFi reads, combining RAFT and Hifiasm improved the assembly contiguity as evidenced by the increased contig NG50 and the number of contigs assembled T2T (Table 2). We observed significant improvements in assembly contiguity when using high-accuracy ultra-long ONT reads as well. We expect that further advances in the accuracy of ONT sequencing and haplotype-aware error-correction algorithms (Stanojevic et al., 2024) would also make ONT Simplex reads amenable to the RAFT-Hifiasm approach. The use of ONT Simplex reads will be useful to achieve a scalable and low-cost method for generating T2T haplotype-resolved genome assemblies. Although we specifically chose to use Hifiasm alongside RAFT, the RAFT approach is easy to implement and can be slotted into any overlap-based assembly algorithm. We do not recommend RAFT if the input read length distribution is already narrow, e.g., using PacBio HiFi reads.

Inspired by the RAFT-Hifiasm approach, Stanojevic et al. (2024) proposed another method of fragmenting long reads into equal-sized sequences and then assembling the fragmented reads. The method reuses the original unfragmented reads in a later step for repeat resolution and graph phasing. Lastly, we note that our read fragmentation approach may appear similar to de Bruijn graph-based methods. However, unlike de Bruijn graph-based methods and similar to the overlap graph-based methods, we use inexact suffix-prefix overlaps of arbitrary lengths to join the nodes. This allows us to choose a long fragment length, e.g., 20 kbp, and not worry about introducing breaks in the graph. In a way, the RAFT approach shares features with both string graph and de Bruijn graph-based methods.

Methods

Counting sequencing outputs containing an assembly gap

In the following, we formally present the details of the CGProb method that calculates the fraction of sequencing outputs which contain an assembly gap in the string graph. First, we will state our simplifying assumptions, define the set of valid sequencing outputs, and characterise those sequencing outputs that are affected by the deletion of contained reads. Subsequently, we will count the number of valid sequencing outputs and the affected sequencing outputs combinatorially. Our method employs conventional techniques from combinatorics, including generating functions (Wilf, 2005) and the inclusion-exclusion principle (Allenby and Slomson, 2010).

Assumptions and notations. We make a few simplifying assumptions to make the theoretical analysis tractable. We consider a genome with a single chromosome and ploidy= 2. We represent each haplotype sequence as a circular string to avoid complications arising from boundaries. Suppose both haplotype sequences have lengths G and differ at a single heterozygous SNP locus. Without loss of generality, we say that the heterozygous locus is at position 1 in the circular diploid genome. We assume that at least one read is sampled from the heterozygous locus on each haplotype. Each sequencing read is a substring of a haplotype sequence. Accordingly, a read is characterised by its haplotype of origin, length, and stop position. An output of a genome sequencing experiment can be represented as a pair of multisets (S_1, S_2), where S_k is a multiset of reads sampled from haplotype $k, k = 1, 2$. Recall that a multiset is a set where each element in the set has an associated multiplicity. Two copies of the same element are identical.

We assume that repeats do not exist in our experimental setup. In other words, if a read's sampling interval overlaps with the heterozygous locus in the genome, then the read has a unique match in its haplotype of origin and no match in the opposite haplotype. Similarly, if a read's sampling interval does not overlap with the heterozygous locus, the read has a unique match in each haplotype. A read's matching interval in a haplotype and all the sub-intervals of this interval are said to be *supported* by that read.

We say that read r_j is *contained* in read r_i if r_j is a proper substring of r_i. For example, reads r_6 and r_7 are contained in read r_5 in Figure 3A. We use N_k to denote the total number of reads on haplotype k. Thus, the total count of reads, denoted by N, is $N_1 + N_2$. Let $N_{k,i}$ denote the number of reads on haplotype k of length i. Note that $\sum_i N_{k,i} = N_k$. Let λ_k be the length of the longest read on haplotype k. We assume that $\lambda_k < G/2, k = 1, 2$, that is, the longest read length on both haplotypes is less than half the genome

length. We will know the values of N_k's, $N_{k,i}$'s, and λ_k's from the user-specified read-length distribution and per-haplotype sequencing depths.

Size of the set of valid sequencing outputs. Once the values of $N_{k,i}$'s are known, we define S to be the set of all valid sequencing outputs consistent with our stated assumptions. We note again that we only consider those sequencing outputs which contain at least one read sampled from each haplotype supporting the heterozygous locus, i.e., the interval $[1, 1]$. For that reason, we compute four quantities T, T_1, T_2, T_{12}:

1. T is the cardinality of the set of those sequencing outputs having $N_{k,i}$ reads of length i on haplotype k for all $i \in [1, \lambda_k]$ and for all $k \in \{1, 2\}$.
2. T_1 is defined similarly to T but with a constraint that no read supports the interval $[1, 1]$ on haplotype 1.
3. T_2 is defined similarly to T but with a constraint that no read supports the interval $[1, 1]$ on haplotype 2.
4. T_{12} is defined similarly to T but with a constraint that no read supports the interval $[1, 1]$ on either haplotype.

Using the principle of inclusion and exclusion, we have $|S| = T - T_1 - T_2 + T_{12}$. We compute T, T_1, T_2, and T_{12} by writing out ordinary generating functions. Ordinary generating functions are a common technique used to count the number of multisets that can be generated using elements with restricted multiplicities (Wilf, 2005). One of the simplest examples of using generating functions is to count the number of sets of size p that can be constructed using n distinct elements ($p \leq n$). In this special case, each element is restricted to multiplicity 1. The corresponding generating function for each element is $(x^0 + x^1)$, where x^0 corresponds to not choosing the element and x^1 stands for choosing the element. Since there are n elements, we multiply $(x^0 + x^1)$ to itself n times. The coefficient of x^p in $(x^0 + x^1)^n$ is $\binom{n}{p}$.

In our setting, the number of reads of length i that stop at position j on haplotype k are restricted by (i) the total number of reads of length i on haplotype k, that is, $N_{k,i}$, and (ii) conditions on whether some reads can stop at a specific position j or not. We use ordinary generating function $f_{i,j,k}(x)$ for reads of length i which stop at position j on haplotype k. The monomial x^n in $f_{i,j,k}(x)$ stands for n identical reads of length i which stop at position j on haplotype k. The coefficient of x^n in $f_{i,j,k}(x)$ is either 1 or 0, which indicates whether or not n identical reads having length i and stopping position j are permitted to exist. For example, the coefficient of $x^{N_{k,i}+1}$ in $f_{i,j,k}(x)$ is 0 because $N_{k,i} + 1$ reads of length i don't exist on haplotype k. The number of multisets of reads of length i on haplotype k, denoted by $\alpha_{k,i}$, is the coefficient of $x^{N_{k,i}}$

in the product $\prod_{j=1}^{G} f_{i,j,k}(x)$. To obtain the total number of multisets of sequencing outputs, we compute $\prod_{k=1}^{2} \prod_{i=1}^{\lambda_k} \alpha_{k,i}$.

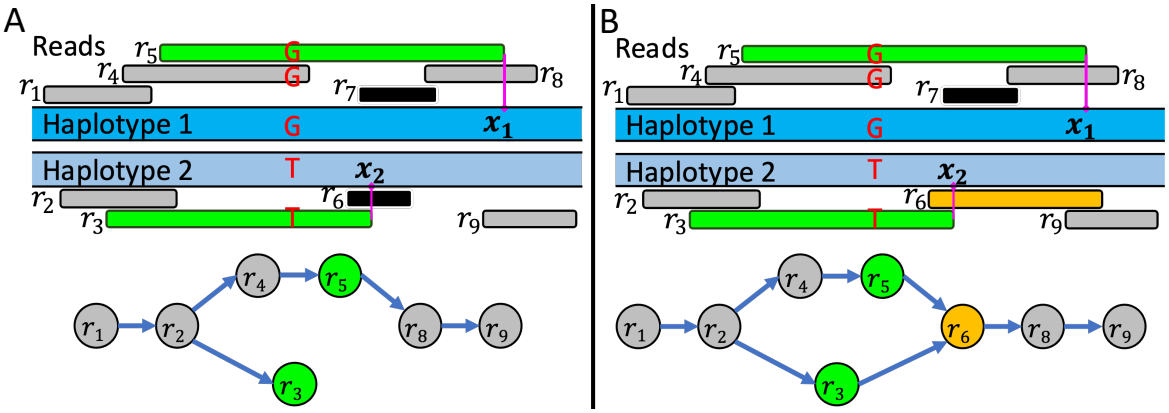

Fig. 3: Illustration of conditions that lead to an assembly gap due to contained read deletion. (A) An example of a sequencing output that is affected by the deletion of contained reads r_6 and r_7. Removing contained reads r_6 and r_7 introduces an assembly gap on haplotype 2. **(B)** An example of a sequencing output where contained read deletion does not introduce an assembly gap. Read r_6 supports the sampling interval of contained read r_7 after its deletion.

To estimate T, we set the ordinary generating functions of reads of length i stopping at position j on haplotype k to $1 + x + x^2 + \cdots + x^{N_{k,i}}$, for $j = 1$ to G. This is because we don't restrict the existence of any number of reads in this case, up to $N_{k,i}$. When estimating T_1, we set the ordinary generating functions of reads of length i stopping at position j on haplotype 1 to be $1 + x + x^2 + \cdots + x^{N_{k,i}}$ provided the read doesn't support the interval $[1, 1]$ on haplotype 1. If it does, then that ordinary generating function is the polynomial x^0, because exactly zero such reads are permitted. Similarly, for estimating T_2, we restrict reads on haplotype 2 from supporting $[1, 1]$. Lastly, for T_{12}, we set the ordinary generating functions of all reads supporting $[1, 1]$ to x^0.

Assembly gap due to contained read deletion. Next, we formally define the occurrence of an assembly gap in a string graph due to contained read deletion. Among all reads on haplotype 1 which support the interval $[1, 1]$, let x_1 be the stop position of those reads which stop closest to position λ_1. Similarly, let x_2 be the stop position of the reads which support $[1, 1]$ and stop closest to position λ_2 on haplotype 2. These reads are shown in green in Figure 3. x_1 and x_2 are well-defined for a given sequencing output. Having identified x_1 and x_2 for a sequencing output $\mathcal{R}$, we say that $\mathcal{R}$ belongs to class (x_1, x_2). Therefore, this assignment partitions the set of all valid sequencing outputs.

Definition 1. *A sequencing output belonging to class* (x_1, x_2) *is said to have an* ***assembly gap due to contained read deletion*** *if the interval* $[\min(x_1, x_2), \min(x_1, x_2) + 1]$ *is originally supported on both haplotypes by some reads, and no longer supported on at least one haplotype by any read after the deletion of contained reads.*

Figure 3A shows an example of a sequencing output where $x_1 > x_2$ and the deletion of contained reads leads to the loss of reads supporting the interval $[x_2, x_2 + 1]$ on haplotype 2. Other gaps in the assembly may also occur naturally due to a lack of coverage on a region in the original sequencing data. These gaps are not considered in our analysis because these are not introduced computationally. On the other hand, an assembly gap in Definition 1 is artificially introduced by a graph simplification heuristic and hinders an assembler from phasing the heterozygous variant. Using Definition 1, we restrict our attention to the interval $[\min(x_1, x_2), \min(x_1, x_2) + 1]$, which is located on the clockwise side of the heterozygous locus in the circular genome. Next, we establish the distinguishing property of the sequencing outputs that are affected by contained read deletion. We will use this property for counting these sequencing outputs.

Lemma 1. *Let* $\mathcal{R}$ *be a sequencing output belonging to class* (x_1, x_2)*.* $\mathcal{R}$ *has an assembly gap due to contained read deletion if and only if* $\mathcal{R}$ *satisfies all the following conditions: (1)* $x_1 \neq x_2$*, (2) At least one read starting in* $[2, \min(x_1, x_2)]$ *on either haplotype stops in* $[1 + \min(x_1, x_2), \max(x_1, x_2)]$*, and (3) No read starting in* $[2, \min(x_1, x_2)]$ *on either haplotype supports* $[1 + \max(x_1, x_2), 1 + \max(x_1, x_2)]$*.*

Proof. Let $\mathcal{R}$ be a sequencing output which satisfies the three conditions. Without loss of generality, assume $x_1 > x_2$. Let $\mathcal{X}$ be the multiset of reads that start in $[2, x_2]$ and stop in $[1 + x_2, x_1]$ on either haplotype. Each read in $\mathcal{X}$ supports the interval $[x_2, x_2 + 1]$ on both haplotypes. The second condition guarantees that $|\mathcal{X}| \geq 1$. However, all reads in $\mathcal{X}$ are contained in some read that supports the interval $[1, x_1]$ on haplotype 1. Accordingly, all reads in $\mathcal{X}$ will be removed by the contained read deletion heuristic. The third condition ensures that no other read in $\mathcal{R}$ supports the interval $[x_2, x_2 + 1]$ on haplotype 2. As a result, an assembly gap due to contained read deletion is guaranteed.

Conversely, suppose $\mathcal{R}$ is a sequencing output which fails to satisfy one of the three conditions. In each case, we prove that an assembly gap due to contained read deletion does not occur in the string graph of $\mathcal{R}$. **Condition (1)**: Suppose $\mathcal{R}$ fails to satisfy the first condition. Therefore, $x_1 = x_2$. In this case, $\min(x_1, x_2) = \max(x_1, x_2) = x_1$. If no read in $\mathcal{R}$ supports $[x_1, x_1 + 1]$ on haplotype 1, then by Definition 1, $\mathcal{R}$ cannot have an assembly gap due to contained read deletion. Accordingly, let us consider the non-empty multiset $\mathcal{Y}$ of the reads that support $[x_1, x_1 + 1]$ on haplotype 1. Let r be a read with the maximum length in $\mathcal{Y}$. By the

definition of x_1 and x_2, read r cannot be contained in a read which supports $[1, 1]$ on haplotype 2. For that reason, any read containing r must support $[x_1, x_1 + 1]$ on haplotype 1. Such a read cannot exist because we selected r with the maximum length from $\mathcal{Y}$. Thus, after deleting all the contained reads, read r will support $[x_1, x_1 + 1]$ on both haplotypes.

Condition (2): Suppose $\mathcal{R}$ satisfies the first condition and does not satisfy the second. Without loss of generality, assume $x_1 > x_2$. We know that no reads start in $[2, x_2]$ on either haplotype and stop in $[1+x_2, x_1]$. Let us analyse the reads supporting the interval $[x_2, x_2 + 1]$. Case (a): There is no read in $\mathcal{R}$ which supports $[x_2, x_2 + 1]$ on haplotype 2 before the deletion of contained reads. Then, $\mathcal{R}$ does not have an assembly gap due to contained read deletion. Case (b): One or more reads in $\mathcal{R}$ support $[x_2, x_2 + 1]$ on haplotype 2. Then we must have a read in $\mathcal{R}$ starting in $[2, x_2]$ and supporting $[1 + x_1, 1 + x_1]$ on both haplotypes. Let r be a read with the maximum length satisfying this condition. Read r cannot be a contained read for the same reason as stated earlier. Thus, r continues to support $[x_2, x_2 + 1]$ on both haplotypes after the deletion of contained reads does not introduce an assembly gap in $\mathcal{R}$.

Condition (3): Suppose $\mathcal{R}$ satisfies the first and second conditions, and fails to satisfy the third. Without loss of generality, assume $x_1 > x_2$. Failure to satisfy condition (3) means that there exist one or more reads $\in \mathcal{R}$ which start in $[2, x_2]$ and support $[x_1 + 1, x_1 + 1]$ on both haplotypes. This is illustrated in Figure 3B. Arguing along the lines of Case (b) of Condition (2), consider a longest read which starts in $[2, x_2]$ and supports $[x_1 + 1, x_1 + 1]$. This is not a contained read and will continue to support $[x_2, x_2 + 1]$ after the deletion of contained reads. This implies that an assembly gap due to contained read deletion does not occur in $\mathcal{R}$.

We will denote M as the set of sequencing outputs containing an assembly gap due to contained read deletion. The ratio $\frac{|M|}{|S|}$ will give us the fraction of sequencing outputs with a user-specified read length distribution containing an assembly gap due to contained read deletion. Let us denote the set of sequencing outputs that belong to class (x_1, x_2) and satisfy Lemma 1 as M_{x_1,x_2}. We describe our method for calculating $|M_{x_1,x_2}|$ in Supplemental Note S1. Using this method, we calculate $|M_{x_1,x_2}|$ for all $x_1 \in [1, \lambda_1]$ and $x_2 \in [1, \lambda_2]$ and add these to obtain $|M|$.

Lemma 2. *The total number of sequencing outputs containing assembly gaps due to contained read deletion is* $|M| = \sum_{x_1=1}^{\lambda_1} \sum_{x_2=1}^{\lambda_2} |M_{x_1,x_2}|$.

We implemented the methods for calculating $|M|$ and $|S|$ in CGProb. The proposed approach is significantly more efficient when compared to a naive method of individually analysing $O(G^N)$ sequencing outputs. The time complexity of our method is polynomially bounded.

CGProb implementation details and experimental setup. We set genome length G to 1 Mbp. CGProb condenses the genome length and the lengths of each read by a user-specified factor (default value = 1000) to speed up the computation. We used arbitrary-precision integer arithmetic (Granlund, 2010) to eliminate numerical error. The read-length distribution can be extracted as a list of distinct read lengths and counts from any long-read sequencing experiment. In our experiments, see Results (subsection "Frequency of observing an assembly gap"), we used three read-length distributions corresponding to PacBio HiFi, ONT Simplex, and ONT Duplex technologies. We obtained these distributions using publicly available datasets. Using the list of distinct read lengths and counts, the value of G, and the per-haplotype sequencing depths, we ran Seqrequester, commit: `31141c1`, (Walenz, 2023) multiple times (fifteen times for normal genome and five times for cancer genome). Each run was used to simulate a revised list of read lengths and counts on each haplotype. CGProb's runtime was 2 hours on a 50x HiFi dataset, 2 hours on a 50x ONT Duplex dataset, and 15 hours on a 50x ONT Simplex dataset on a server with two 24-core Intel Xeon Gold 6248R CPUs. The commands used to run CGProb are described in Supplemental Note S2.

RAFT implementation and benchmarking

RAFT implementation details. In the RAFT-Hifiasm workflow, RAFT uses error-corrected reads, and all-to-all read alignments computed by Hifiasm. In our experiments, we ran Hifiasm (v0.19.8-r603) two times; the first run executed with `-write-ec` parameter returned the error-corrected reads, and the second run executed with `-dbg-ovec` parameter returned all-to-all alignments in the pairwise alignment format (PAF). RAFT places *potential breakpoint markers* on each read. These markers are positioned evenly at intervals of 5 kbp by default. In the highly-repetitive regions of the genome, it is useful to retain long reads to avoid contig breaks. Accordingly, RAFT deletes the markers at those bases which are predicted to be sampled from repetitive regions of the genome. RAFT uses all-to-all alignments to predict repetitive bases in a read. While processing read r, it counts the number of overlapping reads on each base of read r. Assuming cov is the sequencing depth of the input sequencing dataset, and μ is a cutoff parameter (default value = 1.5), RAFT identifies all intervals of length ≥ 5 kbp in read r, where the count exceeds $\mu \cdot cov$ for all bases of the interval. RAFT further extends these intervals by 500 bp on both sides to avoid having a marker very close to a repeat. After deleting the markers from these intervals, RAFT uses the remaining markers for fragmenting

the read into shorter reads. The length of fragmented reads in RAFT is set to 20 kbp as default. Starting from the first base of the read, it finds the first marker located after the first 20k non-repetitive bases and breaks the read at the marker. Similarly, it finds the first marker after the subsequent 20k non-repetitive bases of the read and cuts again. The process continues until it reaches the last base of the read. The user can adjust the algorithm parameters using RAFT's command line interface.

Simulation-based benchmarking procedure. We evaluated the count of coverage gaps caused by contained read deletion using the standard string graph formulation (Myers, 2005), Hifiasm (v0.19.8-r603) (Cheng et al., 2021), and RAFT-Hifiasm. As discussed earlier, Hifiasm rescues a small number of contained reads after building a string graph. The RAFT-Hifiasm method follows a different approach by fragmenting the input reads. We benchmarked the three methods by sampling error-free long-reads from Hifiasm's trio-based assembly of HG002 human genome. We generated five sets of reads using Seqrequester, resembling the read-length distributions of real PacBio HiFi, ONT Simplex, and ONT Duplex sequencing datasets. Given a read set, we identified the set of non-contained reads by generating all-to-all read overlaps using minimap2 (v2.26-r1175) (Li, 2018). First, we identified contained reads entirely encompassed within a longer read with 100% alignment identity based on the minimap2 output. Accordingly, the set of non-contained reads comprised all reads which were not contained.

To evaluate the standard string graph formulation, we aligned the set of non-contained reads to the two HG002 haplotype assemblies, one at a time. We used BEDTools (v2.29.1) (Quinlan and Hall, 2010) to extract all genomic intervals with zero alignment coverage. Next, we excluded the intervals that overlapped with any interval with zero sequencing depth. We also excluded the intervals that included the first or last 25 kbp bases of an HG002 contig to account for edge effects. This left us with a subset of intervals that accurately represented the assembly gaps caused by contained read deletion. We followed the same procedure for Hifiasm but modified the last step because Hifiasm rescues a few contained reads. We ran Hifiasm to compute the rescued reads and aligned the rescued reads to the two HG002 haplotype assemblies. We reported the assembly gaps which remained unresolved. Our benchmarking procedure for RAFT-Hifiasm was similar to Hifiasm, except we considered the set of fragmented reads produced by RAFT instead of the original simulated reads. We recomputed the sets of non-contained reads and rescued reads by following the same procedure.

NaiveCut-Hifiasm. NaiveCut-Hifiasm fragments all error corrected reads in a "repeat-agnostic" fashion, i.e. the reads are fragmented into those reads that RAFT would produce if there were no repetitive regions on the reads. These reads are then assembled using Hifiasm.

Computing assembly quality statistics. We followed Hifiasm's benchmarking procedure (Cheng et al., 2021, 2022) to assess assembly quality. We evaluated the phased assemblies (Table 2, Supplemental Table S3) computed by Hifiasm (v0.19.8-r603), NaiveCut-Hifiasm, and RAFT-Hifiasm methods. A phased assembly comprises two sets of contigs from haplotypes 1 and 2, respectively. In Table 2, we estimated NG50 using QUAST (v5.2.0) (Gurevich et al., 2013) by assuming the size of true assembly as 3.1 Gbp. Additionally, we used: (i) Asmgene (Li, 2018) to compute gene completeness, (ii) Yak (v0.1-r69-dirty) with parental sequencing data to estimate the switch error rate, and (iii) HPRC workflow (`https://github.com/biomonika/HPP/blob/main/assembly/wdl/workflows/assessAsemblyCompletness.wdl`) to estimate the count of T2T-complete contigs. The commands used to run these tools are available in Supplemental Note S3.

Datasets

We used publicly available datasets to obtain read length distributions for the analysis using CGProb. For the PacBio HiFi read length distribution, we used data from SRR10382244, SRR10382245, SRR10382248, and SRR10382249 from the SRA. We obtained the ONT Simplex read length distribution from (`https://s3-us-west-2.amazonaws.com/human-pangenomics/index.html?prefix=working/HPRC_PLUS/HG02080/raw_data/nanopore/HG02080_2.fastq.gz`), and ONT Duplex read length distribution from (`https://human-pangenomics.s3.amazonaws.com/index.html?prefix=submissions/0CB931D5-AE0C-4187-8BD8-B3A9C9BFDADE--UCSC_HG002_R1041_Duplex_Dorado/Dorado_v0.1.1/stereo_duplex/*_stereo_duplex_pass.fastq.gz`) datasets.

For the simulation-based benchmarking procedure, we used a Trio-based assembly of the HG002 human genome by Hifiasm (`ftp://ftp.dfci.harvard.edu/pub/hli/hifiasm-phase/v2/HG002.hifiasm.trio.0.16.1.hap1.fa.gz`, `ftp://ftp.dfci.harvard.edu/pub/hli/hifiasm-phase/v2/HG002.hifiasm.trio.0.16.1.hap2.fa.gz`).

For benchmarking the RAFT-Hifiasm workflow, we used four HG002 read sequencing datasets. Dataset D1 contained PacBio HiFi reads from SRR10382244, SRR10382245, SRR10382248, and SRR10382249. Dataset D2 contained ONT Duplex reads from `https://human-pangenomics.s3.amazonaws.com/index.html?prefix=submissions/0CB931D5-AE0C-4187-8BD8-B3A9C9BFDADE--UCSC_HG002_R1041_Duplex_Dorado/Dorado_v0.1.1/stereo_duplex/*_stereo_duplex_pass.fastq.gz`. Dataset D4 contained ONT high-accuracy ultra-long reads from `https://labs.epi2me.io/gm24385_ncm23_preview/`.

To measure switch error rate in the assemblies, we used complementary Illumina parental data from HG003 and HG004 samples located at (`https://s3-us-west-2.amazonaws.com/human-pangenomics/index.`

html?prefix=NHGRI_UCSC_panel/HG002/hpp_HG002_NA24385_son_v1/parents/ILMN/downsampled/HG003/ and https://s3-us-west-2.amazonaws.com/human-pangenomics/index.html?prefix=NHGRI_UCSC_panel/HG002/hpp_HG002_NA24385_son_v1/parents/ILMN/downsampled/HG004/. Gene completeness metrics were computed using asmgene with cDNA data obtained from http://ftp.ensembl.org/pub/release-102/fasta/homo_sapiens/cdna/Homo_sapiens.GRCh38.cdna.all.fa.gz. For computing assembly QV using Yak (Supplemental Table S3), we used Hi-C data files 'HG002.HiC_2*.fastq.gz' from https://github.com/human-pangenomics/HG002_Data_Freeze_v1.0. GIAB small variant benchmark v4.2.1 was obtained from https://ftp-trace.ncbi.nlm.nih.gov/ReferenceSamples/giab/release/AshkenazimTrio/HG002_NA24385_son/NISTv4.2.1/GRCh38/ for evaluating the variant calls obtained using genome assemblies.

We used ONT Duplex sequencing data for *Solanum lycopersicum* Heinz 1706 (tomato) from https://obj.umiacs.umd.edu/marbl_publications/duplex/index.html. We used the files matching Duplex_*.fastq.gz under the heading "Tomato assemblies and data". The file SRR15243707.1_1.fastq.gz contains PacBio HiFi data. We used it to compute the QV score.

We used ONT Duplex sequencing data for *Zea mays* B73 (maize) from https://obj.umiacs.umd.edu/marbl_publications/duplex/index.html. We used the files R10.4.1_duplex.fastq.gz, maize_duplex.fastq.gz, and PAO10976.fastq.gz under the heading "Maize assemblies and data". We used the PacBio HiFi data in the file named m84006_221229_002525_s1.hifi_reads.fastq.gz to compute the QV score.

Software availability

CGProb is available at https://github.com/at-cg/CGProb. RAFT is available at https://github.com/at-cg/RAFT. CGProb, RAFT, and all custom scripts used to generate the results in this study can be found in Supplemental Code. Other tools used in this study and their software versions are provided in Supplemental Table S5.

Competing interest statement

The authors declare no competing interests.

Acknowledgements

The authors thank Mile Sikic, Sunil Chandran for providing useful feedback. Josipa Lipovac and Prasad Sarashetti tested RAFT code and shared valuable feedback. Haoyu Cheng addressed our concerns re-

garding Hifiasm. We thank the Human Pangenome Reference Consortium for making their sequencing datasets publicly available. This research is supported in part by the funding from the DBT/Wellcome Trust India Alliance (IA/I/23/2/506979) and the National Supercomputing Mission, India under DST/NSM/R&D_HPC_Applications. We also thank the Council of Scientific and Industrial Research (CSIR), Ministry of Science and Technology, India, for the financial support through the Junior Research Fellowship. We used computing resources provided by the National Energy Research Scientific Computing Center (NERSC), USA.

Author contributions

S.S.K., M.B., and C.J. designed the study. S.S.K. and M.B. led the development of CGProb and RAFT, respectively. S.S.K. performed the experiments for analysing assembly gaps. M.B. and S.S.K. assembled sequencing data from HG002 and analysed the assembled contigs. S.S.K. assembled the plant genome sequencing data and analysed the assembled contigs. S.S.K. generated the figures and wrote the manuscript. D.P. and C.J. edited the manuscript.

References

Allenby RB and Slomson A. 2010. *How to count: An introduction to combinatorics*. CRC Press.

Baaijens JA, El Aabidine AZ, Rivals E, and Schönhuth A. 2017. De novo assembly of viral quasispecies using overlap graphs. *Genome research* **27**: 835–848.

Bankevich A, Bzikadze AV, Kolmogorov M, Antipov D, and Pevzner PA. 2022. Multiplex de bruijn graphs enable genome assembly from long, high-fidelity reads. *Nature biotechnology* **40**: 1075–1081.

Cheng H, Asri M, Lucas J, Koren S, and Li H. 2024. Scalable telomere-to-telomere assembly for diploid and polyploid genomes with double graph. *Nature Methods* pp. 1–4.

Cheng H, Concepcion GT, Feng X, Zhang H, and Li H. 2021. Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm. *Nature methods* **18**: 170–175.

Cheng H, Jarvis ED, Fedrigo O, Koepfli KP, Urban L, Gemmell NJ, and Li H. 2022. Haplotype-resolved assembly of diploid genomes without parental data. *Nature Biotechnology* **40**: 1332–1335.

Chin CS, Alexander DH, Marks P, Klammer AA, Drake J, Heiner C, Clum A, Copeland A, Huddleston J, Eichler EE, et al.. 2013. Nonhybrid, finished microbial genome assemblies from long-read smrt sequencing data. *Nature methods* **10**: 563–569.

Chin CS, Peluso P, Sedlazeck FJ, Nattestad M, Concepcion GT, Clum A, Dunn C, O'Malley R, Figueroa-Balderas R, Morales-Cruz A, et al.. 2016. Phased diploid genome assembly with single-molecule real-time sequencing. *Nature methods* **13**: 1050–1054.

Feng X, Cheng H, Portik D, and Li H. 2022. Metagenome assembly of high-fidelity long reads with hifiasm-meta. *Nature Methods* **19**: 671–674.

Granlund T. 2010. The gnu multiple precision arithmetic library. *http://gmplib.org/* .

Gurevich A, Saveliev V, Vyahhi N, and Tesler G. 2013. Quast: quality assessment tool for genome assemblies. *Bioinformatics* **29**: 1072–1075.

Hui J, Shomorony I, Ramchandran K, and Courtade TA. 2016. Overlap-based genome assembly from variable-length reads. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 1018–1022.

Jain C. 2023. Coverage-preserving sparsification of overlap graphs for long-read assembly. *Bioinformatics* **39**: btad124.

Jarvis ED, Formenti G, Rhie A, Guarracino A, Yang C, Wood J, Tracey A, Thibaud-Nissen F, Vollger MR, Porubsky D, et al.. 2022. Semi-automated assembly of high-quality diploid human reference genomes. *Nature* **611**: 519–531.

Koren S, Bao Z, Guarracino A, Ou S, Goodwin S, Jenike KM, Lucas J, McNulty B, Park J, Rautiainen M, et al.. 2024. Gapless assembly of complete human and plant chromosomes using only nanopore sequencing. *bioRxiv* .

Koren S, Walenz BP, Berlin K, Miller JR, Bergman NH, and Phillippy AM. 2017. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome research* **27**: 722–736.

Li H. 2016. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics* **32**: 2103–2110.

Li H. 2018. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* **34**: 3094–3100.

Li H. 2021. Concepts in phased assemblies. `https://lh3.github.io/2021/04/17/concepts-in-phased-assemblies`.

Li H and Durbin R. 2024. Genome assembly in the telomere-to-telomere era. *Nature Reviews Genetics* pp. 1–13.

Logsdon GA, Vollger MR, and Eichler EE. 2020. Long-read human genome sequencing and its applications. *Nature Reviews Genetics* **21**: 597–614.

Myers EW. 1995. Toward simplifying and accurately formulating fragment assembly. *Journal of Computational Biology* **2**: 275–290.

Myers EW. 2005. The fragment assembly string graph. *Bioinformatics* **21**: ii79–ii85.

Nurk S, Koren S, Rhie A, Rautiainen M, Bzikadze AV, Mikheenko A, Vollger MR, Altemose N, Uralsky L, Gershman A, et al.. 2022. The complete sequence of a human genome. *Science* **376**: 44–53.

Quinlan AR and Hall IM. 2010. Bedtools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* **26**: 841–842.

Rautiainen M, Nurk S, Walenz BP, Logsdon GA, Porubsky D, Rhie A, Eichler EE, Phillippy AM, and Koren S. 2023. Telomere-to-telomere assembly of diploid chromosomes with verkko. *Nature Biotechnology* pp. 1–9.

Stanojevic D, Lin D, Florez De Sessions P, and Sikic M. 2024. Telomere-to-telomere phased genome assembly using error-corrected simplex nanopore reads. *bioRxiv* pp. 2024–05.

Tomescu AI and Medvedev P. 2017. Safe and complete contig assembly through omnitigs. *Journal of computational biology* **24**: 590–602.

Vaser R and Šikić M. 2021. Time- and memory-efficient genome assembly with raven. *Nature Computational Science* **1**: 332–336.

Vicedomini R, Quince C, Darling AE, and Chikhi R. 2021. Strainberry: automated strain separation in low-complexity metagenomes using long reads. *Nature Communications* **12**: 4485.

Walenz B. 2023. Seqrequester. `https://github.com/marbl/seqrequester`.

Wenger AM, Peluso P, Rowell WJ, Chang PC, Hall RJ, Concepcion GT, Ebler J, Fungtammasan A, Kolesnikov A, Olson ND, et al.. 2019. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nature biotechnology* **37**: 1155–1162.

Wilf HS. 2005. *generatingfunctionology*. CRC press.

Yang C, Zhou Y, Song Y, Wu D, Zeng Y, Nie L, Liu P, Zhang S, Chen G, Xu J, et al.. 2023. The complete and fully-phased diploid genome of a male han chinese. *Cell Research* pp. 1–17.

Zook JM, Hansen NF, Olson ND, Chapman L, Mullikin JC, Xiao C, Sherry S, Koren S, Phillippy AM, Boutros PC, et al.. 2020. A robust benchmark for detection of germline large deletions and insertions. *Nature biotechnology* **38**: 1347–1355.