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. The string graph (Myers, JCB 1995) is the standard representation of graphs in overlap-based assembly algorithms. The string graph formulation employs graph simplification heuristics, which drastically reduce the count of vertices and edges in a read-overlap graph. One of these heuristics involves removing the reads contained in longer reads. However, this procedure is not guaranteed to be safe. In practice, it occasionally introduces breaks in the graph by removing all reads that cover one or more genome intervals. What factors contribute to such coverage gap events remains poorly understood. In this work, we mathematically derived the probability of a coverage gap near germline and somatic heterozygous variant loci. Our analysis shows that (i) a coverage gap event is two orders of magnitude more likely for Oxford Nanopore reads than PacBio HiFi reads due to differences in their read-length distributions, and (ii) increasing the sequencing depth reduces the probability at an exponential rate. Next, 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 drastically reduces the number of coverage gaps. Using Nanopore Duplex and PacBio HiFi datasets of the HG002 human genome, we achieved a twofold increase in the assembly NG50 and the number of haplotype-resolved T2T contigs compared to Hifiasm.

1 Introduction

Computing high-quality haplotype-resolved *de novo* assemblies remains a principal challenge in genomics research. The T2T assembly of CHM13 human genome [21] is a recent scientific milestone which has inspired further efforts toward achieving T2T assembly of personal diploid human genomes [13, 30]. Third-generation read sequencing technologies, like Pacific Biosciences high-fidelity (PacBio HiFi) reads and Oxford Nanopore Technology (ONT) reads, were instrumental in the construction of the CHM13 reference genome. Currently, PacBio HiFi and ONT Duplex sequencing technologies produce reads that have greater than 10 kbp average length and less than 0.5% error-rates [17, 28].

De novo genome assembly using long reads is most commonly solved using overlap-layout-consensus based methods [20]. The assembly workflow typically involves (i) computing pairwise overlaps between reads, (ii) error-correction of reads, (iii) constructing a read-overlap graph — where the reads are represented as vertices, and suffix-prefix overlaps between reads as directed edges, and (iv) identifying walks in the graph which correspond to contiguous substrings of the genome. A read-overlap graph is initially quite tangled and requires additional graph simplification heuristics to remove redundant or false vertices and edges. Myers's *string graph* formulation [19, 20] has long been the standard choice to build a simplified version of a read-overlap graph [1, 3, 4, 5, 6, 7, 8, 14, 15, 25, 26]. The string graph model was also used by the T2T Consortium to compute the CHM13 human genome assembly [21].

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 in another read [19]. The advantage of the two steps is that they prohibit redundancy, i.e., no two walks in a string graph spell the same sequence [24]. However, prior works have highlighted that the removal of contained reads from the graph is an 'unsafe'

*These authors contributed equally to this work

operation because this heuristic can occasionally disconnect the walks corresponding to true chromosome sequences [3, 8, 11, 12, 21]. The connectivity breaks when all reads that cover one or more genomic intervals are removed. Throughout this paper, we will refer to such intervals as *coverage gaps* in a string graph. The need to remove contained reads is currently a major weakness of the read-overlap-based assembly algorithms [17].

A few approaches have been proposed to tackle the above problem. The algorithms in [11] and [12] 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 [4] 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 [12, 17]. A more recent version of Hifiasm also uses the alignment of ultra-long nanopore reads to a string graph to identify important contained reads [3]. This is a useful approach if ultra-long reads are available. Previous experiments [12] have reported that coverage gaps in a string graph are more likely to occur 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 of the other haplotype.

This paper makes two contributions. Firstly, we mathematically derived a formula to estimate the probability of observing a coverage gap in healthy and cancer genomes. We implemented this calculation in an open-source software CGProb (`https://github.com/at-cg/CGProb`). Given haplotype-specific sequencing depths and a read-length distribution as input, CGProb outputs the probability of a coverage gap event near a germline or a somatic heterozygous variant locus. We studied the probability values in different experimental settings, e.g., with different choices of sequencing technology and sequencing depth. This analysis reveals novel insights into the key contributing factors of coverage gap events.

Second, we developed RAFT (Repeat-Aware Fragmenting Tool, `https://github.com/at-cg/RAFT`) to address the issue of contained read deletion during genome assembly. Conceivably, the fraction of contained reads in a sequencing dataset is roughly determined by its read-length distribution. On the one hand, there is no contained read if all reads have a fixed length (e.g., Illumina reads), whereas an ONT sequencing dataset may have a significant fraction of contained reads due to a wide read-length distribution [18]. RAFT reduces the variation in the read-length distribution by fragmenting long reads into equal-sized shorter reads. The reads predicted to span repetitive regions in the genome are treated differently. RAFT enables high-quality haplotype-resolved assemblies of variable-length long 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.

2 Results

2.1 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. Figure 1A shows an example where a coverage gap event is observed after the contained read r_8 is deleted. In general, this is a random event because its occurrence depends on several parameters, including the sampling positions of reads, genome heterozygosity, read lengths, sequencing coverage, ploidy, etc. As a result, deriving the expected number of coverage gaps in a string graph is challenging.

We developed CGProb to estimate the probability of observing a coverage gap near a heterozygous locus. CGProb expects a read-length distribution and sequencing depths of two haplotypes as input. We made a few simplifying assumptions, such as (i) reads are sampled uniformly at random in each haplotype, (ii) there is one heterozygous SNP in the genome, (iii) reads are error-free, and (iv) each read matches uniquely within its haplotype of origin. The sample space includes all possible multisets of read-sampling intervals that are consistent with the user-provided read-length distribution and haplotype-specific sequencing depths. CGProb estimates the probability by counting those outcomes in the sample space where there is a coverage gap. The theory and implementation details will be formally presented in the Methods section. A naive counting method would involve checking all possible $O(G^N)$ read sampling outcomes individually, where G is

the genome length and N is the number of reads. Instead, we used efficient partitioning of the sample space and the concept of generating functions [29] to calculate the probability in polynomial time (Methods).

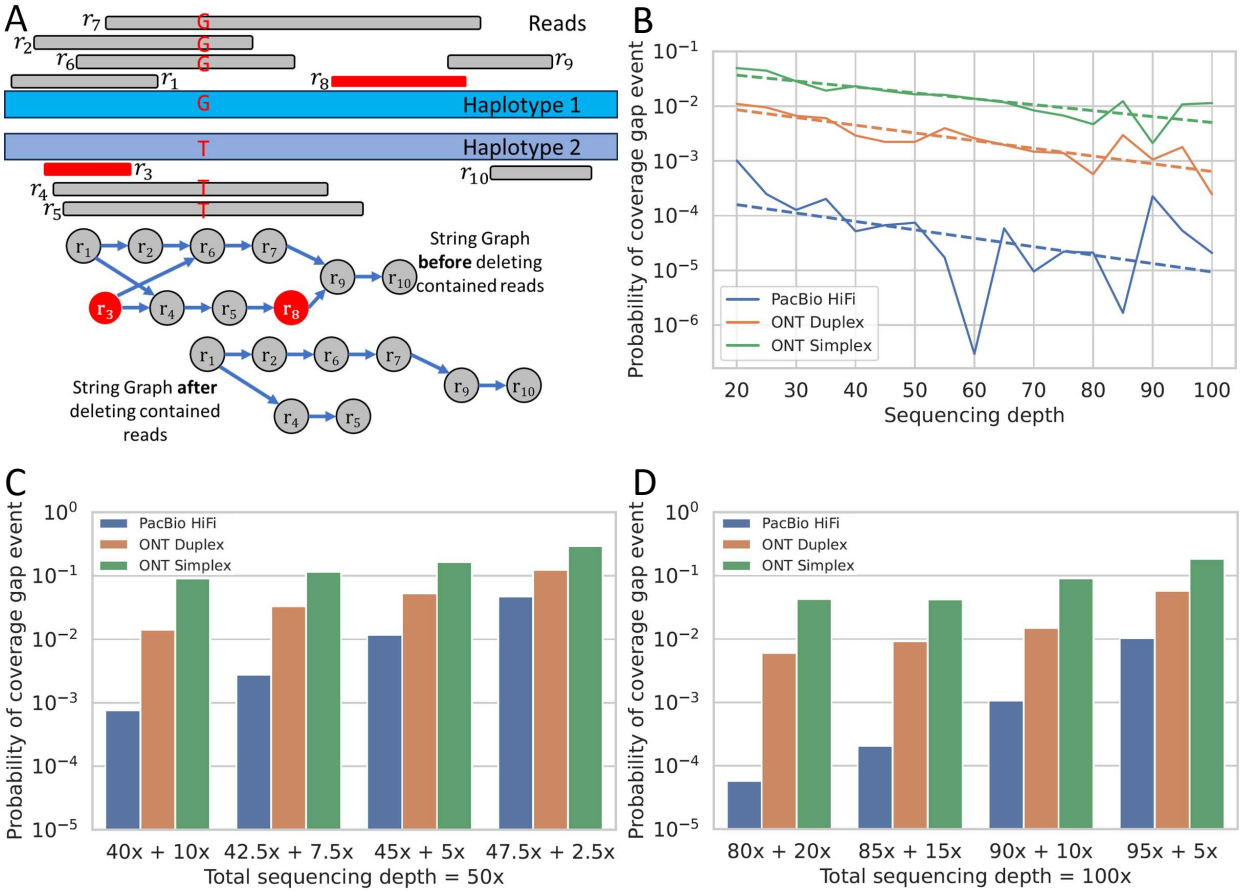

Figure 1: Coverage gaps due to contained read deletion and their probability of occurrence. (A) An example of a coverage gap event caused by the removal of contained reads. The string graph representation excludes contained reads r_3 and r_8. Here read r_3 is contained in read r_1, and read r_8 is contained in read r_7. 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)** Probability of observing a coverage gap. We measured the probability values using three read-length distributions (PacBio HiFi, ONT Duplex, ONT Simplex) at different sequencing depths. Here, we used equal sequencing depths on both haplotypes. **(C)** and **(D)** Probability of observing a coverage gap near a heterozygous allele with uneven sequencing depths on two haplotypes. This scenario models somatic mutation in DNA with variant allele frequency below 0.5. In (C), the total sequencing depth for both haplotypes is 50×; in (D), the total sequencing depth is 100×.

2.2 Probability of observing a coverage gap

We studied the probability of observing a coverage gap event separately for a healthy and cancer genome. In the first scenario, the heterozygous SNP is a germline mutation where we used equal sequencing depths for both haplotypes (paternal and maternal). In the second scenario, the heterozygous SNP is a somatic mutation with variant allele frequency* below 0.5; here, we used uneven sequencing depths for the two haplotypes. We studied the probability estimates in each scenario using the three read-length distributions corresponding to PacBio HiFi, ONT Simplex, and ONT Duplex technologies. These three read-length distributions were obtained using publicly available sequencing data (Methods). As expected, the read-length distribution is

*Variant allele frequency is the fraction of reads supporting a specific DNA variant divided by the overall coverage at that locus.

most skewed in ONT Simplex and least skewed in PacBio HiFi (Supplementary Figure S1). We ran CGProb fifteen times for each input and recorded the average value.

(1) Probability of coverage gap near a germline heterozygous variant locus: We estimated the probability values while varying whole-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 probability of observing a coverage gap using ONT reads compared to PacBio HiFi reads. The results imply that coverage gaps are far more likely to occur when there is high variability in the read lengths. The fraction of reads that are contained will be more if there is higher variability in the read lengths. The same plot also reveals an exponential decrease in the probability with an increase in the sequencing depth. If the sequencing coverage is higher, then the set of reads in a genome interval becomes larger in size which reduces the chance of every read in the set being contained.

(2) Probability of coverage gap near a somatic heterozygous variant locus: We analyzed the probability estimates in a heterogeneous sequencing sample, e.g., a sample with mixed normal and tumour cells. We set total sequencing depths as 50× and 100×. We set the tumor sequencing depths as $5\%, 10\%, 15\%$, and 20% of the total (Figures 1C, 1D). For all three sequencing technologies, we observed that the probability of a coverage gap increases as the tumour sequencing depth decreases. The result implies that the coverage gaps are more likely to occur near somatic genetic variants with low variant allele frequencies. This is also 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. Again, ONT reads are more likely to have a coverage gap compared to PacBio HiFi reads.

2.3 Overview of RAFT

The above analysis clearly indicates that the problem of coverage gaps is much less prevalent in sequencing datasets with narrow read-length distributions. Inspired by these results, we developed RAFT as a practical solution to assemble long-read datasets with wide read-length distributions. The RAFT algorithm fragments long reads into shorter, uniform-length reads while also taking into consideration the potential usefulness of the longer reads in assembling complex repeats. We envision RAFT as a module that can be easily integrated in any existing overlap-layout-consensus-based assembler.

The input to the RAFT algorithm includes error-corrected long reads and the 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 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 prohibit the interval from being fragmented. Such intervals are potentially repetitive and may be necessary to resolve repetitive sequences (Figure 2C). 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). We designed the 'RAFT-Hifiasm' workflow that combines RAFT's ability to manipulate the read-length distribution and Hifiasm's highly efficient error-correction and all-vs-all alignment algorithms [4]. 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. In the end, we assemble the fragmented reads using Hifiasm.

2.4 Evaluation using simulated data

We simulated *error-free* long reads from a publicly available haplotype-resolved HG002 diploid human genome assembly using Seqrequester [27]. We generated 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 (Supplementary Table S1). 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.

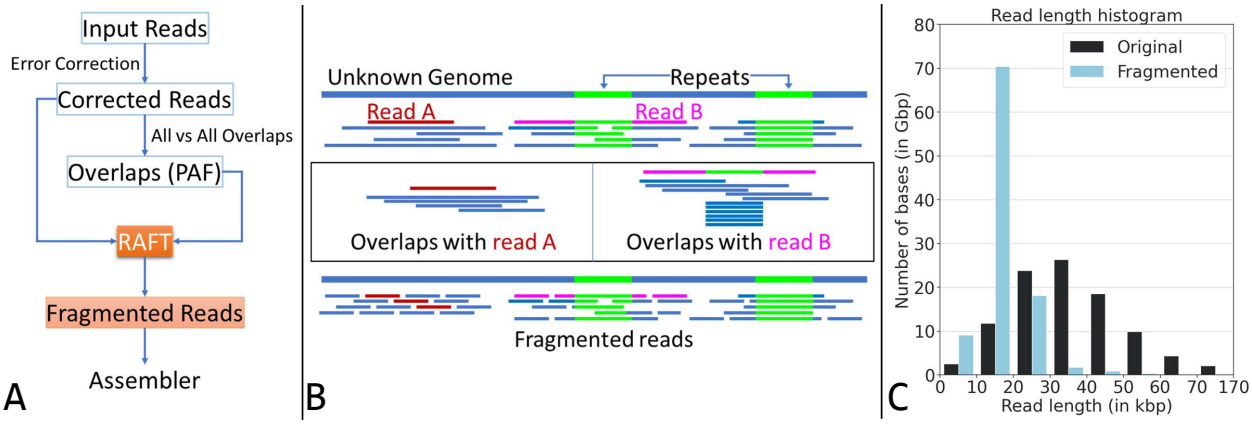

Figure 2: Illustration of the RAFT algorithm and its role in genome assembly. (A) Flowchart of an assembly workflow that uses RAFT. RAFT accepts error-corrected long reads and all-to-all alignment information as input. It produces a revised set of fragmented reads with a narrow read-length distribution. **(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) bridges 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 width of the read-length distribution is significantly reduced by using RAFT. The original dataset contains 3.7 million reads with a skewed read length distribution. After fragmentation, the dataset contains 6.8 million reads.

We tested RAFT-Hifiasm and Hifiasm methods to evaluate their ability to address the issue of coverage gaps. The string graph formulation [19] uses non-contained reads and ignores contained reads. Hifiasm [4] 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. The benefit of using simulated data is that we know the genome sequence from where the reads were sampled. One way to measure the coverage gaps is by aligning the set of reads used in the string graphs to the HG002 genome. Any interval in the genome which has zero read-alignment coverage and non-zero sequencing depth is a coverage 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 coverage gaps by at least an order of magnitude. We eliminated the gaps entirely in two datasets (Table 1). A small number of coverage gaps remain 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. We will demonstrate the impact of this approach on genome assembly in the next section.

		PacBio HiFi	ONT Simplex		ONT Duplex	
		30×	30×	50×	30×	50×
Number of coverage gaps	Standard string graph	5	51	11	30	10
	Hifiasm	3	40	6	17	4
	RAFT-Hifiasm	0	2	1	1	0
Fraction of bases used in graph	Standard string graph	44.42%	23.72%	17.76%	30.76%	22.76%
	Hifiasm	44.42%	23.95%	17.83%	30.77%	22.76%
	RAFT-Hifiasm	72.88%	60.76%	57.01%	65.59%	61.83%

Table 1: Evaluation of RAFT-HiFiasm using simulated data. We show the count of coverage gaps in the string graphs caused by contained read deletion, and the fraction of bases used in the graphs. RAFT reduces the fraction of contained reads by narrowing the read-length distribution.

2.5 Evaluation using real 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 D1 dataset includes 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 down-sampled version of D3 with coverage halved to 34×. The read-length statistics of these datasets are available in Supplementary Table S1. We assembled these datasets using RAFT-Hifiasm and Hifiasm methods to compare their output. The commands and software versions for reproducing the analysis are listed in Supplementary Note S3. We skipped comparison with other recent long-read assemblers such as Verkko [23], and LJA [2]. Verkko is not applicable for datasets D1-D4 because it requires ultra-long reads. Even with ultra-long reads, Verkko produces contigs of comparable sizes and accuracy as Hifiasm [3]. LJA's support for diploid genome assembly was under development at the time of preparing this manuscript.

We expected improvements in the datasets comprising ONT Duplex reads (D2-D4) because these datasets have wide read-length distributions. The results obtained using RAFT-Hifiasm and 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 haplotype-resolved assemblies using datasets D2-D4 as indicated by the NG50 metric and the count of T2T-complete contigs. 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 for datasets D2-D4, the assemblies obtained 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 [31], is provided in Supplementary Tables S2 and S3.

Dataset	**Method**	**Size (Gbp)**	**NG50 (Mbp)**	**Switch error (%)**	**T2T contigs**	**Multicopy genes retained(%)**	**Gene completeness**	
							Complete (%)	**Duplicated (%)**
D1: HiFi (36×)	Hifiasm	3.047/3.051	59.0/45.4	1.08/0.95	0	83.13/76.90	97.38/97.36	0.65/0.67
	RAFT-Hifiasm	3.043/3.070	51.0/45.3	1.08/0.97	1	76.90/84.49	97.28/97.52	0.55/0.68
D2: Duplex (32×)	Hifiasm	3.002/3.054	39.8/29.7	2.38/1.66	2	80.98/81.45	96.46/96.33	1.44/1.56
	RAFT-Hifiasm	3.045/2.996	75.6/61.2	1.80/2.06	6	81.69/80.90	97.32/96.93	0.87/1.16
D3: HiFi (36×)+ Duplex (32×)	Hifiasm	3.141/3.003	38.1/43.9	0.82/1.01	1	83.85/77.38	95.40/95.03	2.65/2.77
	RAFT-Hifiasm	3.040/3.050	89.6/88.9	0.98/1.02	6	81.22/82.57	97.83/96.94	0.59/1.16
D4: HiFi (18×)+ Duplex (16×)	Hifiasm	3.056/2.984	42.6/26.3	1.11/0.98	0	82.41/77.06	96.73/96.99	1.21/0.80
	RAFT-Hifiasm	3.024/3.042	90.9/78.3	1.11/1.15	6	81.14/81.06	97.49/97.32	0.75/0.77

Table 2: Evaluation of the RAFT-Hifiasm workflow for haplotype-resolved human genome assembly. We measured assembly quality statistics separately for both haplotypes. The NG50 for each phased assembly was computed by assuming 3.1 Gbp as the genome length. The software and commands used to measure the assembly statistics are available in Supplementary Note S3.

Hifiasm used 9.5 hours to assemble the largest dataset, D3, on a CPU-based server with 128 cores. In contrast, the RAFT-Hifiasm workflow took 18.5 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 only about 1.5 hours. A tighter software integration of RAFT and Hifiasm in the future may help to avoid redundant steps and optimize the runtime.

3 Discussion

This paper analyses and addresses a longstanding weakness of the string graph formulation [19, 20]. String graphs have been commonly used in several *de novo* genome and metagenome assemblers during the past three decades, but the issue of coverage gaps caused by contained read deletion came to limelight only

recently [3, 12, 17, 21]. 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 [17]. Contained read deletion occasionally leads to loss of useful reads in regions of low heterozygosity of diploid or polyploid genomes and in highly repetitive regions of genomes. In both cases, all reads that support an interval in the genome may be discarded when they are contained in a longer read sampled from a near-identical but different region of the genome.

Using CGProb, we studied how the probability of a coverage gap near a heterozygous locus changes with respect to read-length distributions and sequencing depths. This is the first comprehensive mathematical model for this problem. Our analysis showed that the probability of a coverage gap event is at least one order of magnitude higher in ONT sequencing data compared to PacBio Hifi reads because PacBio Hifi reads have much less variability in read lengths. In both cases, the probability decreases rapidly with an increase in the sequencing coverage. We implemented this calculation in CGProb. CGProb allows users to evaluate the probability in their own datasets and planned sequencing depth. In the future versions of CGProb, we hope to extend this theory to relax a few simplifying assumptions, e.g., to evaluate the probability when ploidy exceeds two, and for genome sequences that are heterozygous at more than one closely-spaced loci.

We also presented RAFT as a solution to address the issue of coverage gaps by fragmenting reads to obtain a more uniform read-length distribution. Using ONT Duplex reads and mixtures of ONT Duplex and PacBio Hifi reads, combining RAFT and Hifiasm improved the assembly contiguity as evidenced by the increased NG50 and number of contigs assembled T2T (Table 2). Future improvements in the accuracy of ONT Simplex sequencing data would also make them amenable to the RAFT-Hifiasm approach. 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. RAFT complements a recent approach [3] of using ultra-long ONT reads to rescue useful contained reads. In RAFT, there is no strict dependency on ultra-long reads. If available, ultra-long ONT reads can also be used in the RAFT-Hifiasm workflow to further improve assembly contiguity and the number of T2T contigs (Supplementary Table S4).

4 Methods

4.1 Calculating the probability of a coverage gap event

In this section, we formally present the details to calculate the probability of observing a coverage gap due to contained read deletion. Our sample space will be all possible read sequencing outputs which may exist under the user-specified per-haplotype sequencing depths and read-length distribution constraints. As expected, the size of the sample space grows rapidly with the number of distinct read lengths and coverage. Simulating and testing each individual read sequencing output for a coverage gap event would be a Sisyphean endeavour. Instead, we identify the exact conditions for observing a coverage gap event in a read sequencing output and count them using generating functions [29] and the principle of inclusion and exclusion.

Assumptions and notations. We make a few simplifying assumptions. We consider a genome with ploidy= 2. We represent each haplotype sequence as a circular string. Suppose both haplotype sequences have lengths G and differ at a single heterozygous SNP locus. 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 $k, k = 1, 2$.

We assume that repeats do not exist in our experimental setup. Accordingly, 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. The loci in a read's matching interval in the genome are said to be *supported* by that read.

We assume that the stop positions of reads are sampled uniformly at random on each haplotype. The length of a read is assumed to be independent of its stop position. We say that read r_2 is contained in read r_1 if r_2 is a proper substring of r_1. We use N_k to denote the total number of reads on haplotype k. 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

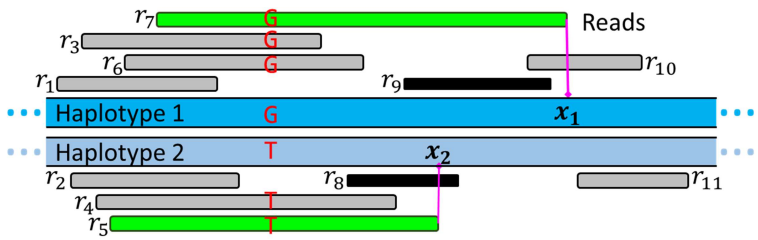

Figure 3: Coverage gap event of type (x_1, x_2). The stop positions of the reads r_7 and r_5 on haplotypes 1 and 2 are x_1 and x_2 respectively. Removing contained reads r_8 and r_9 introduces a coverage gap in the string graph on haplotype 2.

$\lambda_k < G, k = 1, 2$. We will know the values of N_k's, $N_{k,i}$'s, and λ_k's from the user-specified per-haplotype sequencing depths and read-length distribution.

Counting all possible read sequencing outputs. We will compute the probability of observing a coverage gap by counting the number of read sequencing outputs containing a coverage gap event and dividing it by the total number of read sequencing outputs. Recall that n indistinguishable objects can be placed into m distinguishable bins in $\binom{n+m-1}{n}$ ways. The $N_{k,i}$ number of reads of length i on haplotype k are only distinguishable by their stop positions. Treating the stop positions as distinguishable bins, the number of ways in which the $N_{k,i}$ number of reads can be stored in G distinguishable bins is $\binom{G+N_{k,i}-1}{N_{k,i}}$. This is the number of multisets of reads of length i on haplotype k. Thus, the total number of multisets of reads on haplotype k is $\prod_i \binom{G+N_{k,i}-1}{N_{k,i}}$. The total number of read sequencing outputs is $T = \prod_{k=1}^{2} \prod_i \binom{G+N_{k,i}-1}{N_{k,i}}$.

Counting sequencing outputs which have coverage gap. Without loss of generality, we assume that the heterozygous locus is at position 1 in the circular diploid genome. We next consider the distinguishing properties of the read sequencing outputs which have a coverage gap due to contained read deletion. We will count the sequencing outputs which have a coverage gap in haplotype 2 such that the gap occurs in the clockwise direction with respect to the heterozygous locus.

Definition 1. *A read sequencing output has a* ***coverage gap event of type (x_1, x_2)****, where $x_1 > x_2 > 0$, if all of the following conditions are satisfied:*

1. *At least one read supports position 1 on haplotype 1. Among these reads, x_1 is the stop position of those reads which extend the furthest towards position G.*
2. *At least one read supports position 1 on haplotype 2. Among these reads, x_2 is the stop position of those reads which extend the furthest towards position G.*
3. *At least one read starting after position 1 on either haplotype supports position x_2 and stops before or at x_1.*
4. *No read starting in $[2, x_2]$ supports position $x_1 + 1$.*

Definition 2. *A read sequencing output has a coverage gap if it has a coverage gap event of type (x_1, x_2) for some (x_1, x_2).*

Figure 3 gives an example of a read sequencing output with a coverage gap event of type (x_1, x_2). For a given read sequencing output, $1 < x_1 \leq \lambda_1$, and $1 \leq x_2 \leq \min(\lambda_2, x_1 - 1)$. Clearly, a read sequencing output cannot have a coverage gap event of two different types. Our strategy is to count the sequencing outputs which have a coverage gap event of type (x_1, x_2) for each possible value of (x_1, x_2). The sum of these counts will give us the total number of sequencing outputs which have a coverage gap. For a given read sequencing output having a coverage gap of type (x_1, x_2) and $k = 1, 2$, we partition the reads from haplotype k into four classes. Reads in class 1 support the heterozygous locus and stop at x_k. Reads in class 2 support the heterozygous locus and stop before x_k. Reads in class 3(a) start and stop in $[2, x_2]$. Reads in class 3(b) start in $[2, x_1]$ and stop in $[x_2 + 1, x_1]$. Reads in class 4 start in $[x_2 + 1, G]$ and stop in $[x_1 + 1, G]$. For example, the green-coloured reads in Figure 3 are in class 1, and the black-coloured reads are in class 3(b).

Lemma 1. *The read classes 1, 2, 3(a), 3(b), and 4 are mutually exclusive and exhaustive.*

We defer the proofs of Lemmas 1-4 to Supplementary Note S1. Next we define $M_{11}, M_{12}, M_{13}, M_{14}$ as following:

- M_{11} = Number of multisets of reads on haplotype 1 where we permit classes 1 and 3(b) to have zero or more reads.
- M_{12} = Number of multisets of reads on haplotype 1 where we permit class 3(b) to have zero or more reads but force class 1 to have no reads.
- M_{13} = Number of multisets of reads on haplotype 1 where we permit class 1 to have zero or more reads but force class 3(b) to have no reads.
- M_{14} = Number of multisets of reads on haplotype 1 where we force classes 1 and 3(b) to have no reads.

Similarly, the corresponding numbers of multisets on haplotype 2 are denoted using symbols M_{21}, M_{22}, M_{23}, M_{24}. Following the Definition 1, we apply the principle of inclusion and exclusion to count the sequencing outputs which have (i) at least one read in class 1 from haplotype 1, (ii) at least one read in class 1 from haplotype 2, and (iii) at least one read in class 3(b) from either haplotype 1 or 2.

Lemma 2. *The total number of read sequencing outputs with a coverage gap event of type* (x_1, x_2)*, denoted by* M_{x_1,x_2} *equals* $(M_{11} - M_{12})(M_{21} - M_{22}) - (M_{13} - M_{14})(M_{23} - M_{24})$.

As it turns out, the eight quantities $M_{11}, M_{12}, M_{13}, M_{14}, M_{21}, M_{22}, M_{23}, M_{24}$ can be efficiently calculated using ordinary generating functions [29]. First, we write an ordinary generating function $f_{i,j,k}(x)$ representing the number of valid reads of length i stopping at position j on haplotype k. Second, the coefficient of the monomial of degree $N_{k,i}$ in the product $\prod_{j=1}^{G} f_{i,j,k}(x)$ gives the number of multisets of reads of length i on haplotype k. Third, the product of the numbers of multisets of reads of length i on haplotype k (over all possible values of i) gives the number of multisets of reads on haplotype k. We refer the reader to Supplementary Note S1 for details. This analysis yields M_{x_1,x_2}. Any particular read sequencing output can have a coverage gap event of type (x_1, x_2) for at most one pair (x_1, x_2) – which is determined by the reads which support position 1 on both haplotypes. Thus, the following lemma holds.

Lemma 3. *The total number of read sequencing outputs with coverage gap is* $\sum_{x_1=2}^{\lambda_1} \sum_{x_2=1}^{\min(\lambda_2, x_1-1)} M_{x_1,x_2}$.

We implemented the above method in CGProb. The proposed method is significantly more efficient when compared to a naive method of individually analysing $O(G^N)$ read sequencing outputs. The time complexity of our method is polynomially bounded.

Lemma 4. *The runtime of the proposed method for probability estimation is in* $O(\lambda^3 GN \log N)$*, where* $\lambda = \max(\lambda_1, \lambda_2)$.

CGProb implementation details and experimental setup. We set genome length G to 1 Mbp. 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, we used three read-length distributions corresponding to PacBio HiFi, ONT Simplex, and ONT Duplex. We used publicly-available PacBio HiFi (SRR10382244, SRR10382245, SRR10382248, and SRR10382249), ONT Simplex (`https://s3-us-west-2.amazonaws.com/human-pangenomics/working/HPRC_PLUS/HG02080/raw_data/nanopore/HG02080_2.fastq.gz`), and ONT Duplex (`https://humanpangenome.org/data/`) 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`) [27] to obtain a revised list of read lengths and counts on each haplotype. 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 extra-precise 128-bit floating point arithmetic [9] to minimise numerical error. CGProb's runtime was 1.5 hours using 50x HiFi dataset, 5 hours using 50x ONT Simplex dataset, and 0.5 hours using 50x ONT Duplex dataset on a 128-core AMD server.

4.2 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 (v0.19.4-r575) 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 size ≥ 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 [19], Hifiasm (v0.19.4-r575) [4], 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 (`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`). We generated five sets of reads (Table 1) 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) [16]. 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) [22] 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 which included the first or last 25 kbp bases of a HG002 contig to account for edge effects. This left us with a subset of intervals that accurately represented the coverage 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 coverage 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.

Computing assembly quality statistics. We followed Hifiasm's benchmarking procedure [4, 5] to assess assembly quality. We evaluated phased (Table 2, Supplementary Table S3) and primary assemblies (Supplementary Table S2) computed by Hifiasm (v0.19.4-r575) and RAFT-Hifiasm methods. A phased assembly comprises two sets of contigs from haplotypes 1 and 2, respectively. A primary assembly represents a mosaic of the two haplotypes. In Table 2, we estimated NG50 using QUAST (v5.2.0) [10] by assuming the size of true assembly as 3.1 Gbp. Additionally, we used: (i) Asmgene [16] to compute gene completeness, (ii) Yak (v0.1-r69-dirty) and 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 Supplementary Note S3.

Acknowledgements

The authors thank Sunil Chandran, Mile Sikic for providing useful feedback. Josipa Lipovac and Prasad Sarashetti tested RAFT code and shared valuable feedback. Haoyu Cheng addressed our concerns regarding 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 National Supercomputing Mission, India under DST/NSM/ R&D_HPC_Applications and the Science and Engineering Research Board (SERB) under SRG/2021/000044. 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 (JRF). We used computing resources provided by the National Energy Research Scientific Computing Center (NERSC), USA.

References

[1] Jasmijn A Baaijens, Amal Zine El Aabidine, Eric Rivals, and Alexander Schönhuth. "De novo assembly of viral quasispecies using overlap graphs". In: *Genome research* 27.5 (2017), pp. 835–848.

[2] Anton Bankevich, Andrey V Bzikadze, Mikhail Kolmogorov, Dmitry Antipov, and Pavel A Pevzner. "Multiplex de Bruijn graphs enable genome assembly from long, high-fidelity reads". In: *Nature biotechnology* 40.7 (2022), pp. 1075–1081.

[3] Haoyu Cheng, Mobin Asri, Julian Lucas, Sergey Koren, and Heng Li. "Scalable telomere-to-telomere assembly for diploid and polyploid genomes with double graph". In: *arXiv preprint arXiv:2306.03399* (2023).

[4] Haoyu Cheng, Gregory T Concepcion, Xiaowen Feng, Haowen Zhang, and Heng Li. "Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm". In: *Nature methods* 18.2 (2021), pp. 170–175.

[5] Haoyu Cheng, Erich D Jarvis, Olivier Fedrigo, Klaus-Peter Koepfli, Lara Urban, Neil J Gemmell, and Heng Li. "Haplotype-resolved assembly of diploid genomes without parental data". In: *Nature Biotechnology* 40.9 (2022), pp. 1332–1335.

[6] Chen-Shan Chin, David H Alexander, Patrick Marks, Aaron A Klammer, James Drake, Cheryl Heiner, Alicia Clum, Alex Copeland, John Huddleston, Evan E Eichler, et al. "Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data". In: *Nature methods* 10.6 (2013), pp. 563–569.

[7] Chen-Shan Chin, Paul Peluso, Fritz J Sedlazeck, Maria Nattestad, Gregory T Concepcion, Alicia Clum, Christopher Dunn, Ronan O'Malley, Rosa Figueroa-Balderas, Abraham Morales-Cruz, et al. "Phased diploid genome assembly with single-molecule real-time sequencing". In: *Nature methods* 13.12 (2016), pp. 1050–1054.

[8] Xiaowen Feng, Haoyu Cheng, Daniel Portik, and Heng Li. "Metagenome assembly of high-fidelity long reads with hifiasm-meta". In: *Nature Methods* 19.6 (2022), pp. 671–674.

[9] Torbjörn Granlund. "The GNU multiple precision arithmetic library". In: *http://gmplib.org/* (2010).

[10] Alexey Gurevich, Vladislav Saveliev, Nikolay Vyahhi, and Glenn Tesler. "QUAST: quality assessment tool for genome assemblies". In: *Bioinformatics* 29.8 (2013), pp. 1072–1075.

[11] Joseph Hui, Ilan Shomorony, Kannan Ramchandran, and Thomas A. Courtade. "Overlap-based genome assembly from variable-length reads". In: *2016 IEEE International Symposium on Information Theory (ISIT)*. 2016, pp. 1018–1022. DOI: 10.1109/ISIT.2016.7541453.

[12] Chirag Jain. "Coverage-preserving sparsification of overlap graphs for long-read assembly". In: *Bioinformatics* 39.3 (2023), btad124.

[13] Erich D Jarvis, Giulio Formenti, Arang Rhie, Andrea Guarracino, Chentao Yang, Jonathan Wood, Alan Tracey, Francoise Thibaud-Nissen, Mitchell R Vollger, David Porubsky, et al. "Semi-automated assembly of high-quality diploid human reference genomes". In: *Nature* 611.7936 (2022), pp. 519–531.

[14] Sergey Koren, Brian P Walenz, Konstantin Berlin, Jason R Miller, Nicholas H Bergman, and Adam M Phillippy. "Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation". In: *Genome research* 27.5 (2017), pp. 722–736.

[15] Heng Li. "Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences". In: *Bioinformatics* 32.14 (2016), pp. 2103–2110.

[16] Heng Li. "Minimap2: pairwise alignment for nucleotide sequences". In: *Bioinformatics* 34.18 (2018), pp. 3094–3100.

[17] Heng Li and Richard Durbin. "Genome assembly in the telomere-to-telomere era". In: *ArXiv* (2023).

[18] Glennis A Logsdon, Mitchell R Vollger, and Evan E Eichler. "Long-read human genome sequencing and its applications". In: *Nature Reviews Genetics* 21.10 (2020), pp. 597–614.

[19] Eugene W Myers. "The fragment assembly string graph". In: *Bioinformatics* 21.suppl_2 (2005), pp. ii79–ii85.

[20] Eugene W Myers. "Toward simplifying and accurately formulating fragment assembly". In: *Journal of Computational Biology* 2.2 (1995), pp. 275–290.

[21] Sergey Nurk, Sergey Koren, Arang Rhie, Mikko Rautiainen, Andrey V Bzikadze, Alla Mikheenko, Mitchell R Vollger, Nicolas Altemose, Lev Uralsky, Ariel Gershman, et al. "The complete sequence of a human genome". In: *Science* 376.6588 (2022), pp. 44–53.

[22] Aaron R Quinlan and Ira M Hall. "BEDTools: a flexible suite of utilities for comparing genomic features". In: *Bioinformatics* 26.6 (2010), pp. 841–842.

[23] Mikko Rautiainen, Sergey Nurk, Brian P Walenz, Glennis A Logsdon, David Porubsky, Arang Rhie, Evan E Eichler, Adam M Phillippy, and Sergey Koren. "Telomere-to-telomere assembly of diploid chromosomes with Verkko". In: *Nature Biotechnology* (2023), pp. 1–9.

[24] Alexandru I Tomescu and Paul Medvedev. "Safe and complete contig assembly through omnitigs". In: *Journal of computational biology* 24.6 (2017), pp. 590–602.

[25] Robert Vaser and Mile Šikić. "Time- and memory-efficient genome assembly with Raven". In: *Nature Computational Science* 1.5 (2021), pp. 332–336.

[26] Riccardo Vicedomini, Christopher Quince, Aaron E Darling, and Rayan Chikhi. "Strainberry: automated strain separation in low-complexity metagenomes using long reads". In: *Nature Communications* 12.1 (2021), p. 4485.

[27] Brian Walenz. *Seqrequester*. `https://github.com/marbl/seqrequester`. 2023.

[28] Aaron M Wenger, Paul Peluso, William J Rowell, Pi-Chuan Chang, Richard J Hall, Gregory T Concepcion, Jana Ebler, Arkarachai Fungtammasan, Alexey Kolesnikov, Nathan D Olson, et al. "Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome". In: *Nature biotechnology* 37.10 (2019), pp. 1155–1162.

[29] Herbert S Wilf. *generatingfunctionology*. CRC press, 2005.

[30] Chentao Yang, Yang Zhou, Yanni Song, Dongya Wu, Yan Zeng, Lei Nie, Panhong Liu, Shilong Zhang, Guangji Chen, Jinjin Xu, et al. "The complete and fully-phased diploid genome of a male Han Chinese". In: *Cell Research* (2023), pp. 1–17.

[31] Justin M Zook, Nancy F Hansen, Nathan D Olson, Lesley Chapman, James C Mullikin, Chunlin Xiao, Stephen Sherry, Sergey Koren, Adam M Phillippy, Paul C Boutros, et al. "A robust benchmark for detection of germline large deletions and insertions". In: *Nature biotechnology* 38.11 (2020), pp. 1347–1355.