A combinatorial approach for reconstructing rDNA repeats

Frederik Oehl

A thesis presented for the degree of
Master of Science

Algorithmic Bioinformatics
Heinrich Heine University Düsseldorf
Germany
April 21, 2022

Acknowledgements

I would like to thank my supervisors Professor Gunnar W. Klau and Professor Tobias Marschall for their guidance and dedicated support during the preparation and development of this thesis. I would like to thank Sven Schrinner for the many exciting and helpful discussions we had, and for his valuable comments on the thesis. Moreover, I would like to seize the opportunity to thank everyone who gave me intellectual or emotional support during the past two years, as these were not the easiest of times for completing one's study.

Computational infrastructure and support were provided partly by the Centre for Information and Media Technology, and partly by the Algorithmic Bioinformatics chair at Heinrich Heine University Düsseldorf.

Declaration

I hereby confirm that this thesis is my own work, and that I have only used the sources and materials specified in my thesis.

Wuppertal, April 21, 2022

Frederik Oehl

Abstract

The rDNA repeat sites are one of the regions on the human genome where assembly is most difficult. In this thesis, we present a method that reconstructs the individual rDNA repeat copies of human samples. It allows for a study of the variation between the copies from one sample, as well as between different samples. Our method builds on techniques for graph assembly and sequence-to graph alignment, that have recently been developed during the Telomere-to-Telomere (T2T) Consortium's effort to construct a new reference genome, and on the use of ultra-long PacBio HiFi and Oxford Nanopore reads. On an assembly graph that we construct with the help of these methods and technologies, we solve a combinatorial optimization problem. It yields the rDNA repeat copies of a given sample as output. We prove the hardness of this optimization problem by proving the NP -completeness of its corresponding decision problem.

In order to demonstrate the viability of our method, we present assemblies of the rDNA repeat copies from six human samples. One of the samples is CHM13, the basis for the new reference genome that the T2T Consortium recently completed. This allows us to compare the output of our model in detail with the T2T Consortium's assembly of the rDNA repeat sites. The other five samples come from the Human Pangenome Reference Consortium (HPRC). We hope that, in the future, we can improve our model further, and can also develop methods for partitioning the repeats onto different chromosomes and haplotypes.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

The genome of all known life forms contains sequences of rDNA. They code for ribosomal RNA (rRNA), an important constituent of ribosomes [1], molecular machines that are responsible for protein synthesis in the cell. They contribute to a variety of important biological functions, including aging [2]. On the human genome, there exist hundreds of long, highly similar sequences that code for rRNA. These sequences, called rDNA repeats, contain the coding regions for the 18S, 28S and 5.8S rRNA. All rDNA repeat copies in the human genome are located on the short arms of the five acrocentric chromosomes (13,14,15,21,22), where they occur in arrays consisting of dozens of copies. These arrays are referred to as rDNA repeat sites. The total number of copies on the genome varies between different individuals. A study by Malinovskaya et al. [3], conducted on 651 individuals from Moscow and surrounding regions, found between 200 and 711 repeat copies per human sample. A study by Parks et al. [4], conducted on 2546 samples from the 1000 Genomes project, even found the copy number varying between 61 and 1590 per individual.

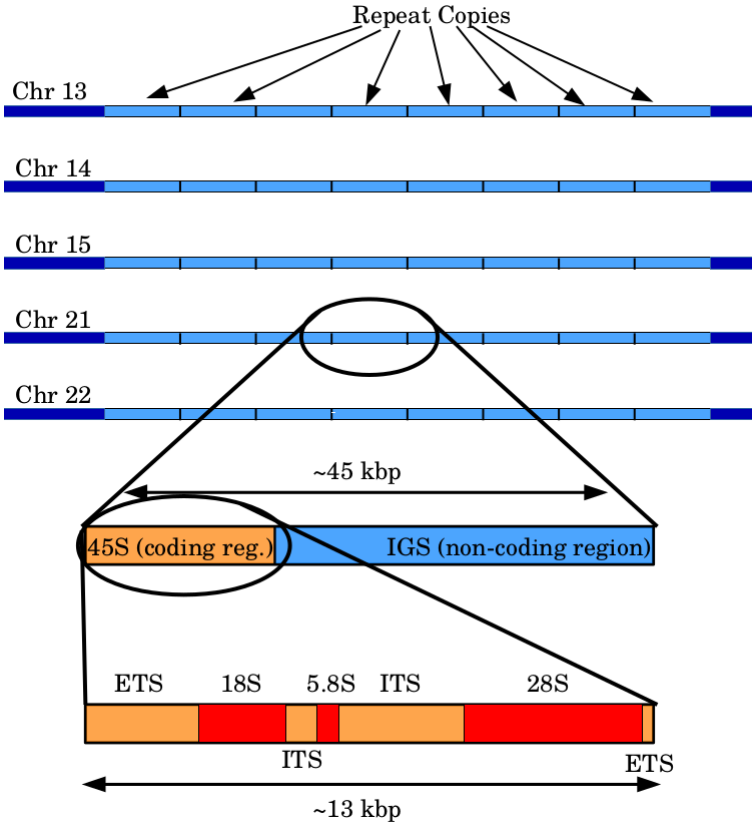

Figure 1: The rDNA repeat sites in the human genome. On the short arms of chromosomes 13,14,15,21 and 22, there are arrays consisting of dozens of repeat copies. Their number can vary strongly between individual human beings, and between the different chromosomes of one human beinng. Each repeat copy consists of a coding region, known as 45S, and a non-coding region, or intergenic spacer. The 45S region consists of the 18S, 5.8S and 28S regions, that code for parts of the ribosome [3], and internal and external transcribed spacers (ITS and ETS).

By default, an individual repeat copy in the human genome consists of a coding region of about 13 kbp length, also known as the 45S sequence, on which the 18S, 28S and 5.8S sequences are located, followed by an intergenetic spacer (IGS) with a length of about 30 kbp [1]. Figure 1 gives a visual representation of the rDNA repeats in the human genome. Contrary to this idealized depiction, however, real rDNA repeat copies are not exactly identical. They are considered similar to each other, but they can differ in a number of ways, such as point mutations, insertions, deletions, or inversions. There is even evidence for large-scale variation between the repeat copies, including insertions and deletions of several thousand base pairs, and inversions of whole copies [5].

Due to their length and similarity, the rDNA repeat sites are one of the sections of the genome that are most difficult to resolve. Until a short time ago, all available types of reads were too short to span an entire repeat copy. This made proper assembly virtually impossible for a long time. Neither the variation between the individual copies, nor their order on the genome could thus be determined. Only recently, as part of the newly published, complete human reference genome assembled by the Telomere-to-Telomere (T2T) Consortium [6], the first reconstruction of the rDNA repeat sites of an individual human being was made available.

In the present, the development of a generic method for precise and fast *de novo* reconstruction of rDNA repeats from any given sample is getting within sight. One reason for this is the invention of new sequencing techniques that lead to new types of reads, especially PacBio HiFi [7] and ONT [8]. Another is the development of new methods and tools for assembly, such as MBG [9] and GraphAligner [10], during the course of the T2T Consortium's assembly of the CHM13 reference genome. Building on these efforts, we propose a combinatorial optimization-based model for reconstructing all individual rDNA repeat copies from given human samples. In this thesis, we present this model in detail, and demonstrate its utility.

We begin by outlining the research context in section 1.2. This includes a description of the T2T Consortium's reconstruction of the CHM13 rDNA repeat sites, as well as a description of approaches similar to ours, that have been used for solving other genome assembly problems. We also state the scope and limitations of our method. In section 2, we describe the method itself. We start with a summary of the algorithms behind MBG and GraphAligner, since these tools are important in the preprocessing of the data we use. We continue with a description of the preprocessing steps in our workflow. Then, we formally introduce the model itself. We prove the NP-hardness of the problem we want to solve by showing the NP-completeness of its corresponding decision problem. Afterwards, we describe methods to compare the output of our model with other reconstructions, and to assess the plausibility of its output. In section 3, we present the results from our experiments with the model. We start by presenting our reconstruction of the repeat copies from CHM13. We evaluate the performance of our model, and compare the assembled repeat copies extensively with the T2T Consortium's assembly. Then,

we move on to present our reconstruction of the repeat copies from five Human Pangenome Reference Consortium (HPRC) samples. In section 4, we evaluate the results from our model, and discuss ideas for future work and an expansion of our method. We end this thesis with a conclusive summary in section 5.

1.2 Background & related work

1.2.1 Telomere-to-Telomere Consortium reference genome

The Telomere-to-Telomere (T2T) Consortium recently finished its reference genome CHM13, which is the first complete reconstruction of a human genome. The respective papers ([6] and [11] among others) have recently been published in *Science*. Compared to GRCh38, the most comprehensive reference genome so far, T2T-CHM13 contains about 200 million bp of newly reconstructed sequence. This includes 81 million bp of new or structurally variant, long, highly similar repeat sequences [11], and, specifically, the rDNA repeat sites. Now follows a summary of the T2T Consortium's reconstruction of the rDNA sites, as described in [6].

As a first step, the authors used MBG to build a De Bruijn graph from PacBio HiFi reads, which have very low error rates despite their length of 13.5 kbp on average. They choose a k-mer size of 3501 for the assembly, which resulted in an output graph with low connectivity. The authors interpreted different components and different clusters within the largest component of the graph as belonging to the five different repeat sites on the chromosomes. They assigned one component or cluster to each of these sites. Based on a k-mer comparison with the graph components, they assigned the HiFi reads to the five distinct sites. Subsequently, the authors assembled new graphs for the five chromosomes with MBG, this time using a much smaller k-mer size of 201. The resulting graphs all show a large-scale circular structure. In order to obtain the arrays of repeat sequences on the chromosomes, the authors included Oxford Nanopore (ONT) reads from the repeat sites into the analysis. These reads can be hundreds of thousands of base pairs long, and therefore span several repeat copies, but they are much more error-prone than the HiFi reads. Like the HiFi reads, the ONT reads were compared to the first graph in order to assign them to the different chromosomes.

The authors then aligned the ONT reads to the five graphs that correspond to the five repeat sites, to trace the rDNA repeat copies in the graphs. The copies thus found were put into clusters ('morphs') based on their similarity, and a consensus sequence was created for each of these morphs. Now, for each morph, the authors estimated a copy number based on the number of matching ONT reads. In total, there are 32 different morphs, that is, 32 unique rDNA repeat copies, that made it into the final reconstruction. The largest copy number estimate for a single morph is 66. This means that, in the final reconstruction, there are 66 copies that correspond to this morph. For all other morphs, this works analogously.

For resolving the order of the repeat copies on the chromosome, the authors built five graphs out of the morphs, each one again representing a chromosome. Then, they aligned the ONT reads to the morph graphs to determine paths that represent the sequence of rDNA repeat copies on the genome. For chromosomes 14 and 22 respectively, the authors' reconstruction contains about 20 identical copies that form the center of the repeat sites, surrounded by other copies that are not repeated as often. For chromosomes 13,15, and 21, the authors are less confident when it comes to the order of the copies, since the structure of some of the morph graphs was still very complex, and the ONT reads were too short to allow building reliable walks in the graphs. For determining the order of the copies on chromosomes 13, 14, and 21, the morphs 'were polished and arranged in consecutive blocks, based on their coverage and order in the graph' [6]. The authors remark that the results for these chromosomes should be treated with caution.

One important aspect of the reconstruction process is that the number of rDNA repeat copies on the two haplotypes of a human genome varies. For their reference genome, however, the authors have constructed only one sequence for each chromosome. As a consequence, not all rDNA repeat copies that they assembled originally, were included in the final genome. In the end, the authors selected those repeat copies that were included in the reference genome based on, among other reasons, their similarity to the canonical rDNA unit KY962518.1.

Our own method builds upon the procedures for assembly and sequence-to-graph-alignment developed during the course of the T2T Consortium's reconstruction of CHM13. What is new to our approach, however, is that it yields a mathematical model for turning read data from any given sample, human or even non-human, into optimal repeat copies. We thus present a generic method for rDNA repeat reconstruction, in contrast to a handcrafted approach fine-tuned to a single dataset. Note that the model presented in this thesis only aims to reconstruct the individual rDNA repeat copies, allowing for the study of their variation in a single sample as well as between different samples. Resolving the order of the repeat copies and, finally, haplotyping the copies, are the next steps on the way.

1.2.2 Related assembly methods

In its most basic outlines, our model is about selecting optimal paths on an assembly graph, that are supposed to represent the repeat copies. In the preprocessing, we use MBG to construct an assembly graph from all HiFi reads from a given sample, that map to the rDNA repeat sites. Each node in this graph corresponds to a sequence of base pairs, and comes with a coverage value that denotes how often the sequence appears in the input data. Since we talk about a graph with collapsed unitigs, these values can represent averages. We use GraphAligner to align a set of very long, but error-prone ONT reads from the repeat sites to the graph. Now,

our main goal is to pick a fixed number of these ONT read alignments, that explain the coverage of the nodes as well as possible. This means, each selected alignment explains a certain amount of coverage on each node that it contains, and we want to pick k alignments such that as little coverage as possible is left unexplained. For each selected alignment, we then infer the corresponding repeat copy by concatenating the sequences from the nodes the alignment contains.

There are some instances where similar approaches have been applied successfully in the context of genome assembly. Baaijens et al. [12] use a procedure for assembling viral quasispecies, meaning mutant strains that belong to a single species of virus, that is based on optimization on a variation graph. From a set of input contigs, the authors construct a directed variation graph with a source node s and a sink node t through multiple sequence alignment. Each node in the variation graph corresponds to a sequence of base pairs. For every node, the authors compute an estimated coverage value. They proceed to construct s-t paths that form potential virus haplotypes. For this, they make use of the fact that each of the input contigs corresponds to a subset of nodes in the variation graph. This enables them to concatenate overlapping contigs to form maximal-length paths (meaning s-t paths) on the variation graph. The authors enumerate all possible maximal-length paths that can be formed through contig concatenation. Since, in principle, an exponential number of possible contig concatenations exists, the number of possible paths can become very large. Later, Baaijens et al. improved the path enumeration step by developing a network flow approach for estimating contig abundances, that forms the basis for constructing the concatenated paths [13]. When the possible paths are generated, the authors assign an optimal multiplicity to all of them. 'Optimal' means that the coverage of the nodes is explained as good as possible through the multiplicities of the paths. In the end, the authors pick all paths that have a multiplicity above a cutoff that can be chosen freely.
Other, somewhat similar models relate to solving the MINIMUM PATH COVER problem. For example, Rizzi et al. [14] present a genome assembly-related application of this problem. Given a directed graph, MINIMUM PATH COVER asks for the minimum number of paths, such that each node in the graph is covered. On DAGs, the problem can be solved in polynomial time.

What sets our model apart from all previous approaches we have knowledge of, is its level of generality. We work with an undirected graph and assume no source and target node. The graph can have a complex structure where it is not trivially visible what *could* count as a potential repeat copy, even though we attempt to work with graphs that are as clean and structured as possible. The graph can also have unconnected components. In fact, once we have aligned the ONT reads to the graph, our model becomes more similar to SET COVER than to a classical graph problem. In section 2.4, we will thus prove the problem's hardness by reducing from a variant of SET MULTICOVER. Another point is that we make extensive use of the new generation of long reads. Aligning the long ONT reads to nodes generated from

accurate HiFi reads enables us to generate a not-too-large set of full, potential repeat copies, from which we can then select the optimal ones. We hope that the flexibility of our model, combined with the availabilty of the latest generation of long reads, enables us to meaningfully reconstruct a large number of long and potentially variant repeats, as they occur on the rDNA sites.

1.3 Limitations of the model

The model that forms the basis of this thesis aims at reconstructing the individual repeat copies. In other words, we focus on showing the *variation* that is present in the hundreds of rDNA copies on a human genome. This leaves open the problem of resolving the *order* of the repeat copies on the genome. We do not yet have developed a formal method to solve this problem. However, we will present an idea for ordering the copies, that we consider promising, in section 4.2. Another challenge that we do not cover is haplotyping the repeat copies, that is, finding out whether two given repeat copies are located on the same haplotype of a chromosome. This problem is closely linked to resolving the order of copies on the genome. A good method for haplotyping likely presupposes a good method for ordering. Thus, we consider haplotyping the copies to be the third, and final, step towards complete reconstruction.

2 Method

This section entails a complete description of the methods we employ to reconstruct the rDNA repeat copies. We start by giving an overview of the external tools we use in our workflow. We explain the tools MBG [9] and GraphAligner [10] in some detail due to their importance. Next, we describe the preprocessing steps that are necessary to bring the input data into a form that we can process with our model. The methods we use for the preprocessing were largely developed by the T2T Consortium during the assembly of CHM13 [6]. After having outlined the preprocessing steps, we formally introduce our model for determining the optimal rDNA repeat copies. We prove that the problem we thereby solve is NP-hard. Finally, we describe methods for comparing our output with other reconstructions, especially the rDNA repeat copies from T2T-CHM13, and for checking our output for plausibility.

2.1 External tools

We make use of a number of tools to process the data we examine, and to visualize the output of our model. The most important ones are: MBG for the assembly of de Bruijn Graphs from input sequences, GraphAligner for aligning reads to to sequence graphs, Bandage [15] for visualizing sequence graphs, minimap2 [16] for mapping reads or the output of our model to reference sequences, IGV (Integrative Genomics Viewer) [17] for visualizing the output of our model, and Samtools [18] for various kinds of data processing in our workflow, such as transforming FASTA files into BAM or SAM files, or vice versa. Out of these, we will explain MBG and GraphAligner in some detail below. This is because we use them for crucial preprocessing steps, and they are necessray to bring the input data into a form that our model can process. Thus, it is necessary to understand the broad outlines of how they work algorithmically, and what kind of output they produce.

Note that we assume that MBG and GraphAligner take two types of reads as input: MBG works with PacBio HiFi reads, while GraphAligner takes ONT reads. This is not mandatory for the two tools, but for our purpose, we do not have to mind about other types of reads. We will come back to the reads in section 2.2. For now, it suffices to know that HiFi reads are long and have a very low error rate compared to their length of, on average, 13.5 kbp, while ONT reads have even greater length, but also a much higher error rate.

2.1.1 MBG

In order to construct the assembly graph from PacBio HiFi reads, we use MBG, which is a tool for assembling de Bruijn Graphs from long reads with a low error rate, especially PacBio HiFi reads. It was also used in the T2T Consortium's effort to assemble the CHM13 genome. We will now give a brief description of its method. For details, see the supplementary material of [9].

As input, MBG takes a set of reads, a k-mer size k, a minimizer window size w, and k-mer and unitig abundance cutoffs a and u. In the first step of the assembly process, MBG compresses all homopolymer runs in the input reads to length 1. That means, if the reads contain runs of the same character, like, for example, 'AAAA', they are collapsed into one single character ('A' in this case). The reason is that most errors in HiFi reads concern the length of homopolymer runs. Next, a hash value is assigned to all k-mers in the reads. For the assembly, MBG employs only a subset of the k-mers, known as minimizers. They are selected in the following way: For each window of size w, only the k-mer with the lexicographically smallest hash value is chosen. In each read, we start with the window $[0 \dots w-1]$ and select the respective minimizer, then continue with $[1 \dots w]$, $[2 \dots w+1]$, and so forth. In order to save space, a hash function is applied to the k-mers, compressing them into 128-bit integers. Now, an assembly graph is constructed, where the 128-bit hashes form the nodes. Edges are added between all nodes that correspond to hashes that are adjacent in the reads. Paths that do not branch are collapsed into single nodes, called unitigs. Edges that correspond to sequences that have an abundance smaller than a, meaning they occur less than a times in the reads, and unitigs with an abundance smaller than u, are deleted from the graph. The remaining nodes are then changed back to sequences of base pairs, and the homopolymer runs are inserted into those sequences, resulting in the completed assembly graph. As output, MBG then produces a GFA [19] file containing the nodes and edges of the graph, and the nodes' corresponding sequences. For each node, MBG outputs a coverage value denoting the abundance of the sequence in the read data. Since many nodes are actually collapsed unitigs, this must be interpreted as an average value.

Bluntification. One important, additional assembly step we have to consider is called bluntification. This term refers to the removal of overlaps between the nodes in the assembly graph, subject to the condition that the walks in the original assembly graph are preserved. As input, we have an assembly graph, where pairs of sequences that correspond to pairs of adjacent nodes overlap. As output, we get a *blunted* graph, meaning that each pair of adjacent nodes corresponds to two adjacent sequences, but the sequences contain no overlapping characters. In addition, all walks that are contained in the input graph have to be contained in the output graph, and vice versa.

Literature that describes the exact implementation of bluntification in MBG is scarce, but we want to give at least a description of the principle involved. In the following, we will thus give a brief summary of the bluntification algorithm designed by Eizenga et al. [20], in order to illustrate the procedure. Note that Eizenga et al. implemented their algorithm under the name 'GetBlunted'.

The authors point out that, in order to create a blunted graph from a non-blunted assembly graph, it is sometimes necessary to duplicate nodes, respectively their sequences. Thus, they seek to transform the graph using a minimal amount of sequence duplications. For this, they

exploit a special property of complete bipartite subgraphs in the assembly graph: for adjacent nodes that belong to two partitions of a complete bipartite subgraph, or biclique, with the edges between the partitions marking the adjacencies, the overlaps can be merged transitively. This means that, for each pair of sequences that corresponds to two nodes, the overlapping affix is removed from the sequence on one of the nodes. No duplication of sequences is necessary. In order to maximize the number of overlaps that can be removed by transitive merging, the graph has to be partitioned into a minimum number of bicliques. This problem, referred to as BICLIQUE COVER, is NP-hard. By employing reduction techniques and using beneficial properties of the assembly graph, bluntification with a minimum number of sequence duplications is still often possible in acceptable time.
In general, the takeaway is that bluntification is possible under the guarantee that important structural properties in the graph remain unchanged, but can come at the price of redundancy of sequences.

2.1.2 GraphAligner

For aligning ONT reads to the blunted assembly graph, we use the tool GraphAligner. From the alignments, respectively a subset of best-fitting alignments, we later select the paths that represent the optimal repeat copies. Now, we give a brief description of the way GraphAligner functions.

GraphAligner takes as input a set of reads and a bidirected assembly graph, where the nodes correspond to sequences of base pairs. The assembly graph is usually stored in a GFA file, as produced by MBG. As output, GraphAligner produces a GAF [21] file containing all approximate alignments of the reads to the graph that have sufficient quality. The read alignment is conducted via a complex seed-and-extend procedure that involves lots of techniques for speedup, and adapts banded Dynamic Programming for approximate sequence-to-graph alignment. For each read we want to align, the tool first searches for exact matches (*seeds*) between subsequences from the read, and substrings from the sequences that correspond to graph nodes. By default, the search for seeds is conducted through the comparison of minimizers from reads and graph nodes. Alternative methods, namely search based on maximal unique matches (MUMs), or maximal exact matches (MEMs), can also be employed. Once the the tool has found seeds for a read, the next step is to extend them through approximate matching. Extension is conducted through a modified version of the Needleman-Wunsch-Algorithm [22] for global alignment via Dynamic Programming (DP). In order to speed up the alignment, a cutoff is set, and only the fraction of the DP matrix where the cutoff is not surpassed is calculated. This procedure is called banded alignment, or banded DP [23] [24]. We will come back to the technique in section 2.5. In order to adapt banded alignment to the special requirements of sequence-to-graph alignment, the algorithm of GraphAligner involves a number of extensions of the method. For example, in order to explore different neighboring nodes,

several bands that take up different paths of the DP matrix are calculated. Also, the size of the band, meaning the size of the part of each row in the DP matrix that is actually calculated, is changed dynamically during the alignment.

Finally, we should note that GraphAligner tests for alignment of reads to the graph node sequences that are found in the input file, but also to the reverse complements of these graph node sequences. DNA always comes in two strands. These are complementary to each other, such that 'A' matches to 'T', 'G' matches to 'C', but run in reverse order. For the sequence 'AGAG', the reverse complement would thus be 'CTCT'. If a read matches to the node sequence as it is found in the input file, this is marked by a '>' in the alignment file. If the read matches to the reverse complement of the sequence in the input file, this is marked by '<'.

2.2 Preprocessing

In order to run our model, we first need to bring the input data into an appropriate form. In the following, we will describe how we get from the input, consisting of PacBio HiFi reads and ONT reads, to the blunted assembly graph with aligned reads, on which we run the model. This process involves the tools MBG and GraphAligner presented above. Altogether, the preprocessing steps draw heavily on procedures and tools developed by the T2T Consortium for the CHM13 assembly (see the section on related work as well as [6]). This includes the alignment of long ONT reads into the assembly graph in order to identify potential paths in the graph. Without this work, as well as the progress in long-read sequencing over the last years, our method would not be feasible.

As input, we have two types of long reads: PacBio HiFi and Oxford Nanopore (ONT). These two types have different, and, to some degree, complementary properties that we are going to make use of. The HiFi reads combine a sizeable length of 13.5 kbp on average with a very low error rate of 0.2% [7]. As mentioned above, most of the errors concern the length of homopolymer runs. The greatest strength of the ONT reads, on the other hand, is their length. They frequently exceed 100 kbp, and can be up to a million bp long [8]. This means that a large number of ONT reads from a sample that maps to the rDNA repeat sites are long enough to cover a whole repeat copy with its length of about 45 kbp. However, they have an error rate of about 15% [6], which is a lot higher than in the HiFi reads.

For obtaining the reads that map to the rDNA repeat sites from a given human genome, we look at complete HiFi and ONT read sets for the human sample under study. We map these reads to a canonical rDNA unit (we use KY962518.1[1]), and keep only those that match approximately. When we have obtained the reads, we assemble a blunted sequence graph from all the HiFi reads using MBG. All the repeat copies we later select are paths in this graph. For the graph assembly, there are two things we have to pay attention to: One is k-mer size, and the other is the abundance cutoff for k-mers and unitigs. Regarding k-mers, a large size leads to low connectivity of the graph, while a short size leads to an increase in the number of nodes, edges, and cycles in the graph. Regarding k-mer and unitig abundance, a high cutoff leads to loss of potentially important nodes, while a low cutoff leads to the inclusion of erroneous k-mers into the graph, and to a very large number of nodes and edges. This also means that memory use and runtime increases when the cutoff is set too low. In order to find a good k-mer size for assembling the graph, we tested a large number of different sizes on HiFi reads from T2T-CHM13. In the end, a k-mer size of 351 gave the best compromise between graph connectivity and efficiency. Note that this is a large k-mer size by traditional standards, but

[1] *https://www.ncbi.nlm.nih.gov/nuccore/KY962518.1*

the low error-rate of HiFi reads also allows for the use of k-mer sizes of several thousands of base pairs in some contexts. Regarding abundance, we test a number of different cutoffs for each sample under study, and then settle for the one that represents the best compromise on this specific graph.

As for the structure of the graph, our goal is to work with as few a priori assumptions as possible. We want to be able to process complex input from largely unknown regions of the genome, and therefore want to avoid dependency on assumptions about how these regions must look like if we want to produce meaningful output. Thus, we do not define start or end nodes on the graph, nor assume a direction of the edges, as a condition for running our model. We treat the graph as undirected and otherwise take it as it is outputted by MBG. There is only one exception: We do have to get rid of the graph's large-scale circular structure. This feature of the graph is produced by the fact that the rDNA repeat copies form tandem arrays on the human genome. Since we want each path we later select to correspond to exactly one repeat copy, and the ONT reads are often longer than one copy, we need to modify the graph structure, such that the paths we select do not correspond to two or more consecutive repeat copies. For this, we first align the canonical rDNA unit to the graph. In the area where beginning and end of the alignment are located, we then remove edges until there is no short path between the beginning and the end of the alignment left. Note that this is the only manual intervention into the output of MBG that is necessary. If there is future work on this project, it will involve thinking about a more elegant, generic method for either disposing of the circular structure, or working with it.

The next step is to align the ONT reads to the graph, for which we use GraphAligner with default settings. All alignments are paths over a subset of the graph nodes. The sequence of base pairs that corresponds to an alignment is the concatenation of the sequences that correspond to the graph nodes. For each node sequence, the GAF file contains information on whether the sequence as it is stored in the GFA file, or its reverse complement, is part of the alignment. All alignments are undirected, meaning there are two possible ways to traverse the path in the graph. By aligning the ONT reads to the graph built from HiFi data, we combine the length of the ONT reads and the precision of the HiFi reads, in order to obtain paths that potentially represent single rDNA copies. For each ONT read, we only keep the alignment with the lowest error rate. Before applying the model, we further restrict the set of alignments we select from by keeping only those alignments with suitable length. For example, this can refer to all alignments that differ at most 5%, or 10%, in length from the canonical rDNA unit. Figure 2 gives a visual overview of all the preprocessing steps.

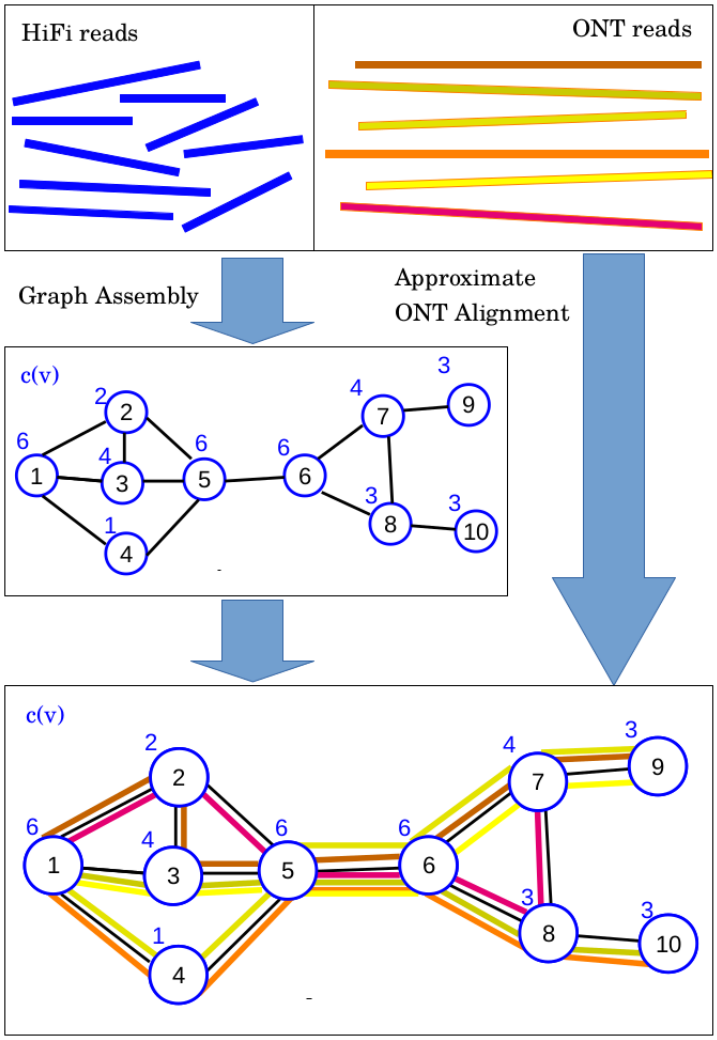

Figure 2: A visualization of the preprocessing for our model. As input, we have PacBio HiFi and ONT reads (top row). From the HiFi reads, we assemble a blunted graph, where each node corresponds to a sequence of base pairs (middle row). For each node, we have a value $c(v)$ that gives an estimate of the abundance of the corresponding sequence in the input data. The blue numbers mark this value. When we have constructed the graph, we align the ONT reads to it, such that each ONT read corresponds to a path in the graph (bottom row). For example, the bright yellow ONT read from the top right image corresponds to the node path $1-3-5-6-7-9$ in the image at the bottom.

2.3 Repeat selection model

Following the preprocessing of the input data, we now introduce the model for finding the optimal set of rDNA repeat copies. First of all, we are given an undirected graph $G = (V, E)$. Each $v \in V$ corresponds to a sequence over the alphabet $\{A, C, G, T\}$, which we are going to call $sequence(v)$. Also, we have two functions $c : V \to \mathbb{R}$ and $w : V \to \mathbb{R}$, that assign a *coverage value* $c(v)$ and a *weight* $w(v)$ to each node. With regard to the output from the preprocessing steps, we interpret the blunted assembly graph that MBG outputs as G, and, for each node in the assembly graph, we interpret the corresponding sequence of base pairs as $sequence(v)$. We interpret the abundance value, or coverage value, MBG outputs for the sequence belonging to a node v as $c(v)$. It refers to how often the sequence occurs in the read data, or how many times it is *covered* by the reads. Since, as described above, we deal with average values, this value is not necessarily an integer, hence $c(v) \in \mathbb{R}$. Through w, we assign a weight to each v based on the length of $sequence(v)$. As a default, we set $w(v) = \log(len(sequence(v)))$.
Furthermore, we are given a multiset $R_{Aln} = \{ra_1, ra_2, ..., ra_n\}$, where each element ra_i is a set of nodes that form a path in G. We interpret the set of ONT read alignments from the preprocessing as R_{Aln}. Each path ra_i represents a potential rDNA repeat copy. Note that R_{Aln} is a multiset, so two paths ra_i, ra_j can be identical. If two identical paths are selected, they count as two independent repeat copies.

As our goal is to find the k rDNA repeat copies present in the sample, we want to select $k \in \mathbb{N}$ paths from R_{Aln} that represent these copies. The set of selected paths we call R_{Opt}. We thus have to determine the optimization criterion or criteria based on which we select $R_{Opt} \subseteq R_{Aln}$, and we have to determine our k.

Regarding the optimization criterion, we want to optimally explain the summed coverage values $c(v)$ of all the sequences that correspond to the nodes v, comparable to what Baaijens et al. [12] do on the viral quasispecies graphs (section 1.2.2). The basic idea is that, since the value $c(v)$ comes from the number of times $sequence(v)$ occurs in the reads, we can use it to estimate how often $sequence(v)$ must occur in the actual repeat copies. We assume that there is an approximately proportional relationship between $c(v)$ and the number of times $sequence(v)$ occurs in the repeat copies. We thus introduce the proportionality factor $c_{avg} \in \mathbb{R}$. It indicates which value of $c(v)$ we expect to be equivalent to one occurrence of $sequence(v)$ in the actual repeats. For example, $c_{avg} = 10$ expresses that, if $c(v) = 10$ for a node v, we expect $sequence(v)$ to occur once in the repeat copies. Note that c_{avg} is an approximate value, and for any single node v, the coverage $c(v)$ is likely not an exact multiple of c_{avg}. However, we assume that, if we choose c_{avg} aptly, the relation holds on average. One possibility for choosing c_{avg} is to try different values and use binary search to find the one that leads to the minimal objective value. Also, the average read coverage in the sample we examine can serve as an orientation point.

We interpret c_{avg} as the amount of coverage that one selected path ra_i *explains* on each node v that is contained in ra_i. This leads us to the value $pathcov(v)$. It denotes the number of paths from R_{Opt} that contain v, multiplied by c_{avg}. Formally, $pathcov(v) = c_{avg} \cdot |\{ra \in R_{Opt} | v \in ra\}|$.
An example might serve to illustrate what we do with the $pathcov$ value. Imagine a node v with a coverage value $c(v) = 50$. Imagine further that c_{avg} is set to 10. Now, if five paths from R_{Opt} contain v, we have $pathcov(v) = 50$. Thus, a coverage of 50 counts as *explained* for v. Since $c(v) = 50$, we have explained all coverage on v. In case only four paths from R_{Opt} contain v, we have $pathcov(v) = 40$, and thus, the coverage $c(v)$ on v is *under-explained* by 10. Conversely, if six paths contain the node, we have *over-explained* its coverage by 10.
Since we do not wish to differ between over- and under-explanation, we can express the relation between $c(v)$ and $pathcov(v)$ for any node v in terms of the absolute difference $|c(v) - pathcov(v)|$ between the two values. In the minimization function, we multiply this term by $w(v)$. This way, we can weigh the nodes, and it becomes more important to explain coverage on some nodes, than on others. If we set $w(v) = \log(len(sequence(v)))$, this means nodes with a long corresponding sequence have more weight than nodes with a short corresponding sequence.

Regarding the number of paths to select k, we want a value that is identical to (or a good estimator of) the number of rDNA repeat copies in the genome under study. Thus, in order to derive k, we look at the set of PacBio HiFi reads. We divide the number of reads that map to the canonical rDNA unit by the number of reads from the whole genome in order to estimate the percentage of the genome that is made up of rDNA repeat copies. Since we know the length of the human genome, as base pair counts for GRCh38 and CHM13 are available online in the NCBI database[2], we can now estimate the total length of the rDNA regions in base pairs. And since we also know the average length of a single rDNA repeat unit, we can estimate the number of repeat copies that fits best to our length estimate. Note that, for now, we do not take ploidy into account. As is the case in haploid reference genomes, such as T2T-CHM13, we treat each repeat copy as if it existed only once. The problem of ploidy is a future challenge that is connected to the problem of haplotyping the copies.

Now, we have all the ingredients to define Problem 1: OPTIMAL REPEAT SELECTION. In the next chapter, we will prove the hardness of this problem by giving an NP-completeness proof for its corresponding decision problem. In order to solve the problem, we formulate it as an Integer Linear Program (ILP). The objective function is convex, but can be translated into linear form, enabling us to use Integer Linear Programming to tackle the problem. The set R_{Opt} that the model yields as output represents the set of rDNA repeat copies. We translate them back into

[2] *www.ncbi.nlm.nih.gov/assembly/GCA_000001405.29 and www.ncbi.nlm.nih.gov/assembly/GCA_009914755.4*

Problem 1.

	OPTIMAL REPEAT SELECTION
Input:	An undirected graph $G = (V, E)$. A multiset of reads $R_{Aln} = \{ra_1, ra_2, ..., ra_n\}$, where each read is a path in G. A value $c(v) \in \mathbb{R}$ for each $v \in V$. A constant $c_{avg} \in \mathbb{R}$. A weight $w(v) \in \mathbb{R}$ for each v. A fixed value $k \in N$, denoting the number of paths to select.
Output:	A subset $R_{Opt} \subseteq R_{Aln}$ with $\lvert R_{Opt} \rvert = k$, such that $\sum_{v \in V} \lvert c(v) - pathcov(v) \rvert \cdot w(v)$ is minimized. For each node $v \in V$, $pathcov(v) = c_{avg} \cdot \lvert \{ra \in R_{Opt} \lvert v \in ra\} \rvert$.

sequences of base pairs by concatenating the sequences that correspond to the nodes. Based on the information on sequence orientation from the GAF file, we decide for each sequence whether to pick the sequence as it is stored in the GFA file, or its reverse complement. Figure 3 gives a visual example for the complete process of picking rDNA repeat copies through solving OPTIMAL REPEAT SELECTION.

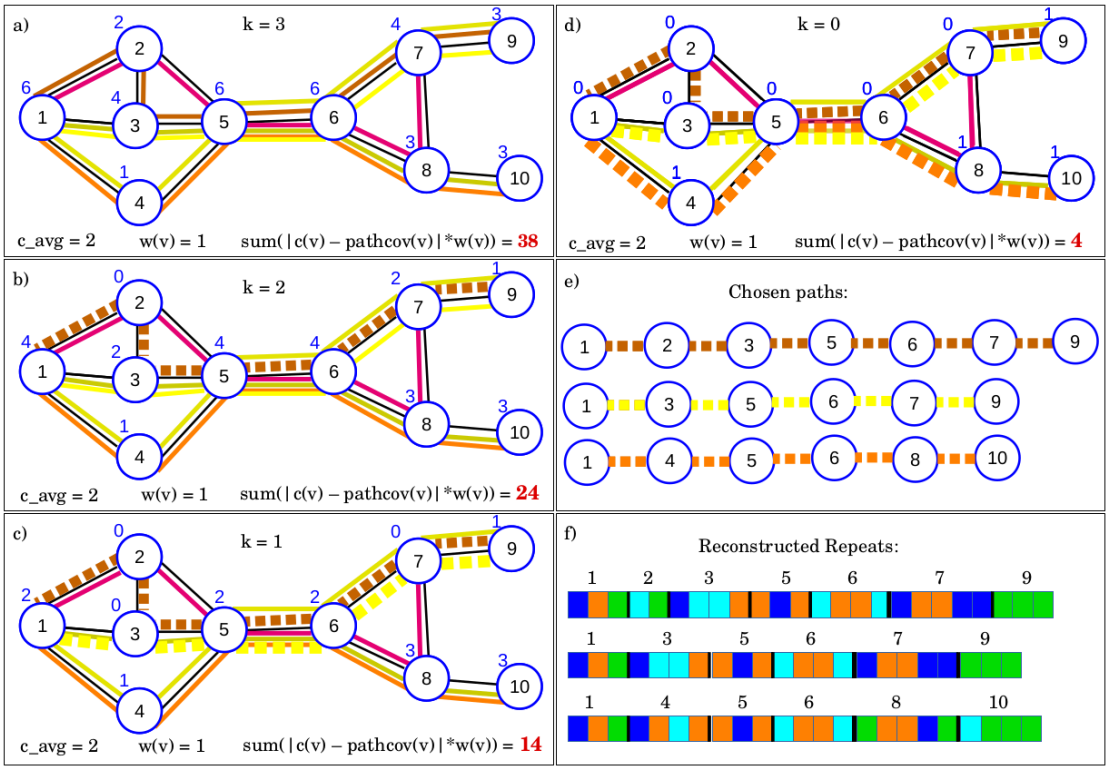

Figure 3: Solving OPTIMAL REPEAT SELECTION on the example instance known from figure 2. In **a)**, we have the graph, with the ONT read alignments marked in different colors. The blue numbers above each node are the values $c(v) - pathcov(v)$. For a), the values are identical to $c(v)$. The amount of coverage that one path explains per node, c_{avg}, is set to 2. The weight $w(v)$ is set to 1. We want to select $k = 3$ paths that minimize the objective function. Before we select any path, the value of the objective function equals $\sum_{v \in V} c(v) \cdot w(v)$. In **b)**, **c)** and **d)**, we always add one path to our selection. Note that, in this figure, we split the selection step into three different subfigures for illustrational purposes. In reality, the problem is not solved iteratively, but globally by an ILP solver. A selected path is marked by a fat, dotted line. In each subfigure, the value k above the graph shows the number of paths we still have to select. The current value of the objective function is always shown in the bottom right corner. In the end, the objective function has a value of 4, which is the minimal value we can achieve with the available paths. In **e)**, we have the 3 optimal paths that have been selected. Each of the nodes corresponds to a sequence of base pairs. In **f)**, we have the actual repeat copies, derived from the paths. The colors denote different base pairs, while the numbers denote the nodes to which the subsequences correspond. The black lines denote the boundaries of the sequences for the single nodes.

2.4 Hardness proof

For proving the hardness of the given problem, we prove the NP-completeness of the corresponding decision problem. In a decision problem, we do not ask for an optimal solution, but instead, we ask whether a solution exists that is smaller than a given value $\ell \in \mathbb{N}$. We define the decision problem REPEAT SELECTION as follows:

Problem 2.

	REPEAT SELECTION
Input:	An undirected graph $G = (V, E)$. A multiset of reads $R_{Aln} = \{ra_1, ra_2, ..., ra_n\}$, where each read is a path in G. A value $c(v) \in \mathbb{R}$ for each $v \in V$. A constant $c_{avg} \in \mathbb{R}$. A weight $w(v) \in \mathbb{R}$ for each v. Two numbers $k \in N$ and $\ell \in N$.
Question:	Is there a subset $R_{Opt} \subseteq R_{Aln}$ with $\lvert R_{opt}\rvert = k$, such that $\sum_{v \in V} \lvert c(v) - pathcov(v)\rvert \cdot w(v) \leq \ell$? For each node $v \in V$, $pathcov(v) = c_{avg} \cdot \lvert\{ra \in R_{Opt} \lvert v \in ra\}\rvert$.

In order to proof the NP-completeness of the problem, we need to show that REPEAT SELECTION $\in$ NP, and that REPEAT SELECTION is NP-hard.

Lemma 1: REPEAT SELECTION $\in$ NP.

Proof Given a certificate that an instance of REPEAT SELECTION is a YES instance, we must be able to check in polynomial time whether the certificate is valid. For an instance of REPEAT SELECTION, a certificate is a multiset of k reads from R_{Aln}. We validate it as follows: For each node $v \in V$, we count the number of times it occurs in the reads from the certificate. We multiply the counts with c_{avg} to get $pathcov(v)$ for each v. We then plug the $pathcov(v)$ values for v into the formula $\sum_{v \in V} \lvert c(v) - pathcov(v)\rvert \cdot w(v)$, and check whether it sums up to at most ℓ. This procedure can be executed in polynomial time: The counting step is in $O(\lvert R_{Aln}\rvert \cdot \lvert V\rvert)$, calculating $pathcov(v)$ for all $v \in V$ is in $O(\lvert V\rvert)$, and computing $\sum_{v \in V} \lvert c(v) - pathcov(v)\rvert \cdot w(v)$ is in $O(\lvert V\rvert)$ as well. It follows that REPEAT SELECTION $\in$ NP. $\square$

In order to proof the NP-hardness of REPEAT SELECTION, we show that SET MULTICOVER WITH MULTIPLICITY CONSTRAINTS $\leq_m^p$ REPEAT SELECTION.

Problem 3. The SET MULTICOVER WITH MULTIPLICITY CONSTRAINTS (SMCwMC) problem is defined by Hua et al. [25] as follows:

> In the set multicover problem, we are given a universe N of n elements and a family of sets $F = \{S_1, ..., S_{|F|}\}$ where each S_i is a subset of N, and we need to find a minimum cardinality sub-family $F' \subseteq F$ such that each element $i \in N$ is covered b_i integral number of times. [...] Note that in order to minimize the total number of picked sets, each set [...] can be chosen a number of times. Here if we further require that each set [...] can be chosen at most a specified number of times, the SMC problem becomes the SMC with multiplicity constraints problem.

For the decision variant of SET MULTICOVER WITH MULTIPLICITY CONSTRAINTS, we are thus given a universe N, consisting of n elements, and a demand $b_i \in \mathbb{N}$ for each element $i \in N$. We have a family of sets F, where each $S_j \in F$ is a subset of N, and set a multiplicity constraint m_j for each S_j. In addition, we have a value $k \in \mathbb{N}_0$. Now, we ask: Can we select $F' \subseteq F$ with $|F'| \leq k$, such that all elements $i \in N$ are covered by the sets in F' at least b_i times?
Note that we interpret the problem in the following way: If we select a set S_j multiple times, each selection adds 1 to the size of F'. The reason is: If multiple selection of S_j did not add to F', the solution for SMC with $b_i \in \mathbb{N}$ would be the same as when all b_i were set to 1, as long as we do not include the multiplicity constraints. In this case, we could just look for a subset of F that covers each element from N, and then multiply the sets as often as we need to fulfill the b_i, with no penalty. This would imply that SMC for arbitrary b_i is trivially identical to SET COVER. Thus, we say that if we select a set S_j multiple times, it adds to the size of F' each time.
SMCwMC is a generalization of the NP-hard [26] problem SET COVER. It is easy to see that the hardness of SET COVER implies hardness of SMCwMC, as any instance of SET COVER reduces to an instance of SMCwMC where the demand b_i is set to 1 for each $i \in N$, and the multiplicity constraint m_j for each set S_j is 1 as well.

Our goal now is to show that SMCwMC reduces to REPEAT SELECTION. We are given an instance I of SMCwMC, consisting of a universe $N = \{1, ..., n\}$ with demands b_i for each $i \in N$, a family of sets $F = \{S_1, ..., S_{|F|}\}$ with multiplicity constraints m_j for each S_j, and a value k. We want to transform it into an instance I' of REPEAT SELECTION with a value $k' = k$ denoting the number of reads we select, and a threshold ℓ we want to underbid, such that SMCwMC(I,k) = TRUE $\iff$ REPEAT SELECTION(I',k',ℓ) = TRUE.
There are two main difficulties. The first is that, in order to fulfill the demands, each element $i \in N$ from I has to be covered *at least* b_i times. In contrast, in order to solve I', we compute an absolute value for each node v. Thus, one cannot straightforwardly take a node $v_i \in V$ and

its value $c(v_i)$ as a substitute for an $i \in N$ and its demand b_i. The second difficulty is that, for I, we look for a solution that consists of *at most* k sets from F, while for I', we look for a solution that consist of *exactly* k reads.
The ideas for solving these two problems are the following: For each $i \in N$ with coverage demand b_i, we create a node v_i with $c(v_i) = b_i$ and a second node v_i' with a very large coverage value $c(v_i')$. These nodes are always contained in the exact same reads. This means that, while coverages greater than b_i increase the penalty for node v_i, they still decrease the penalty for v_i'. The summed penalty for both nodes decreases until a coverage of b_i is reached, and stays the same for coverages greater than b_i. Figure 4 at the end of this section provides a visualization for how this works.
In order to provide that k' is exactly k, we introduce k empty dummy reads. We can represent them as a single dummy read $r_{dummy} = \{\emptyset\}$ with a multiplicity of k'. Thus, if the solution contains less than k' reads, we can simply add dummy reads, such that we have a solution of size exactly k'.

Lemma 2: REPEAT SELECTION is NP-hard.

Based on the ideas described above, we construct I' in the following way: For each element $i \in N$, we create two nodes v_i and v_i'. We set $c(v_i) = b_i$ and $c(v_i') = \sum_{S_j \in F} m_j$, where m_j is the maximal allowed multiplicity for S_j. This means $c(v_i')$ equals the number of non-empty reads in R_{Aln}, including their multiplicities. Thus, no node can possibly be covered by more than $c(v_i')$ reads. Now, for each set S_j, we create a read. We want to be able to select the read as many times as the multiplicity constraint allows the selection of S_j in I. Note that it takes only $log(m_j)$ bits to encode each multiplicity m_j, so generating m_j identical reads would require an exponential number of bits relative to the number of bits we need to store the multiplicity constraints. Since we want a polynomial-time reduction, we have to avoid this. In order to do so, we just assign the multiplicities to the corresponding reads. This means we store the reads' identical copies implicitly. For each element $i \in S_j$, the read that corresponds to S_j contains v_i and v_i'. In addition to the reads that correspond to the sets $S_j \in F$, we create an empty dummy read with a multiplicity of $k' = k$, and add it to our set of reads. We set $c_{avg} = 1$ and $w = 1$. This ensures that for each v or v', $pathcov(v)$ equals exactly the number of covering reads. Finally, we set $\ell = \sum_{v_i, v_i'} c(v_i') - c(v_i)$.

Now, we have to show that SMCwMC(I,k) = TRUE $\iff$ REPEAT SELECTION(I',k',ℓ) = TRUE holds.

"$\Longrightarrow$":
Since we assume SMCwMC(I,k) = TRUE, we know that, for I, we we have a set F' of at most size k, such that all elements $i \in N$ are covered at least b_i times. Now, for I', we select the reads that correspond to the sets in F'. That means, if a set $S_j \in F'$ has a multiplicity of p, we

also select the corresponding read p times. As a consequence, all nodes v_i and v_i' are covered at least b_i times. For each v_i and v_i', we know they occur in exactly the same reads, and are covered the same number of times. Now, thanks to the additional nodes v_i', we have a solution that is $\leq \ell$. There are two cases to consider:

In the case that an element $i \in N$ from I is covered exactly b_i times, both of the corresponding nodes in I' are also covered exactly $b_i = c(v_i)$ times. In other words, we know that in this case, $pathcov(v_i) = pathcov(v_i') = c(v_i)$. It follows that $|c(v_i) - pathcov(v_i)| = 0$, and $|c(v_i') - pathcov(v_i')| = |c(v_i') - c(v_i)| = c(v_i') - c(v_i)$. The penalty for both nodes thus sums up to $c(v_i') - c(v)$. Figure 4c provides an example for this. The rule holds for all pairs v_i, v_i'. Thus, we know that if all elements in I are covered exactly b_i times, $\sum_{v_i,v_i'} c(v_i') - c(v_i)$ is the total penalty in I'. This is exactly ℓ.

In the case that an element $i \in N$ from I is covered more than b_i times, we know that $pathcov(v_i) > c(v_i)$ holds for the corresponding v_i. We also know that, since $c(v_i')$ is at least as big as the number of all reads that can possibly be selected, $pathcov(v_i') = pathcov(v_i) \leq c(v_i')$. From $pathcov(v_i) > c(v_i)$ follows that $|c(v_i) - pathcov(v_i)| = pathcov(v_i) - c(v_i)$. From $pathcov(v_i') = pathcov(v_i) \leq c(v_i')$ follows that $|c(v_i') - pathcov(v_i')| = c(v_i') - pathcov(v)$. Thus, if we sum up the penalties for the two nodes, we get $c(v_i') - c(v_i)$ again. Figure 4d provides an example for this. Now we know that for each element in I that is covered more than b_i times, we get a penalty of $c(v_i') - c(v_i)$ in I'.

It follows that, for each element $i \in N$ from I that is covered at least b_i times, the summed penalty for the two corresponding nodes is $c(v_i') - c(v_i)$. Thus, if SMCwMC(I,k) = TRUE, $\sum_{v_i,v_i'} c(v_i') - c(v_i)$ is the total penalty for I'. This is exactly ℓ. If the solution for SMCwMC(I,k) required less than k sets, we can add the empty dummy read for I' with a multiplicty such that the solution for REPEAT SELECTION(I',k',ℓ) has exactly $k' = k$ reads. We have thus shown "$\Longrightarrow$".

"$\Longleftarrow$":

Since we assume REPEAT SELECTION(I',k',ℓ) = TRUE, we know that, for I', we have $k' \leq k$ non-empty reads that together cover the nodes such that the total remaining penalty is at most $\sum_{v_i,v_i'} c(v_i') - c(v_i)$. Since, for two nodes v_i and v_i', the summed penalty cannot become smaller than $c(v') - c(v)$, we know that we have a total penalty of exactly $\sum_{v_i,v_i'} c(v_i') - c(v_i)$. This, however, can only be the case if each node v_i is covered at least $c(v_i) = b_i$ times. The reason is that, if $pathcov(v_i) = pathcov(v_i') < c(v)$ for nodes v_i, v_i', then $|c(v_i) - pathcov(v_i)| + |c(v_i') - pathcov(v_i')| > c(v_i') - c(v)$. Thus, if a node v_i is covered less than $c(v_i) = b_i$ times, the summed penalty for v_i and v_i' is always greater than $c(v_i') - c(v)$. Figures 4a and 4b exemplify this. We already know the summed penalty for v_i and v_i' is *exactly* $c(v') - c(v)$ for each pair of nodes where $pathcov(v_i) \geq c(v)$, and cannot get smaller. This means that as soon as the summed penalty for one pair of nodes v_i, v_i' is greater than $c(v_i') - c(v_i)$, the total penalty for I' is always greater than $\sum_{v_i,v_i'} c(v_i') - c(v_i) = \ell$. It follows that if REPEAT SELECTION(I',k',ℓ) = TRUE, each element $i \in N$ from I must be covered at least b_i times. We have shown "$\Longleftarrow$".

We now know that SMCwMC reduces to REPEAT SELECTION. We still need to prove that the reduction is possible in polynomial time. Creating v_i and v_i' from all $i \in N$ is linear to the number of elements in N, and thus in $O(N)$. Setting the $c(v_i)$ and $c(v_i')$ values is linear to the number of elements in N as well. Creating the reads from the sets S_j in F could in principle take exponential time, since, for each set S_j, we need as many reads as the multiplicity constraint for S_j allows. These could be exponentially many. Thus, we only create one read for each S_j, and remember its multiplicity m_j. This implicit creation of reads takes linear time compared to the number of elements in F, and is thus in $O(|F|)$. The same holds true for creating the dummy reads. Again, there could be exponentially many of these. But if we just create one dummy read and store its multiplicity, we only need constant time.
Since these are all the necessary steps to construct I' from I, the reduction takes only polynomial time. It follows that SMCwMC $\leq_m^p$ REPEAT SELECTION, and therefore, we know that REPEAT SELECTION is NP-hard. □

Now, we can prove the main result.

Theorem 1: REPEAT SELECTION is NP-complete.

Proof REPEAT SELECTION $\in$ NP and REPEAT SELECTION is NP-hard $\implies$ REPEAT SELECTION is NP-complete. □

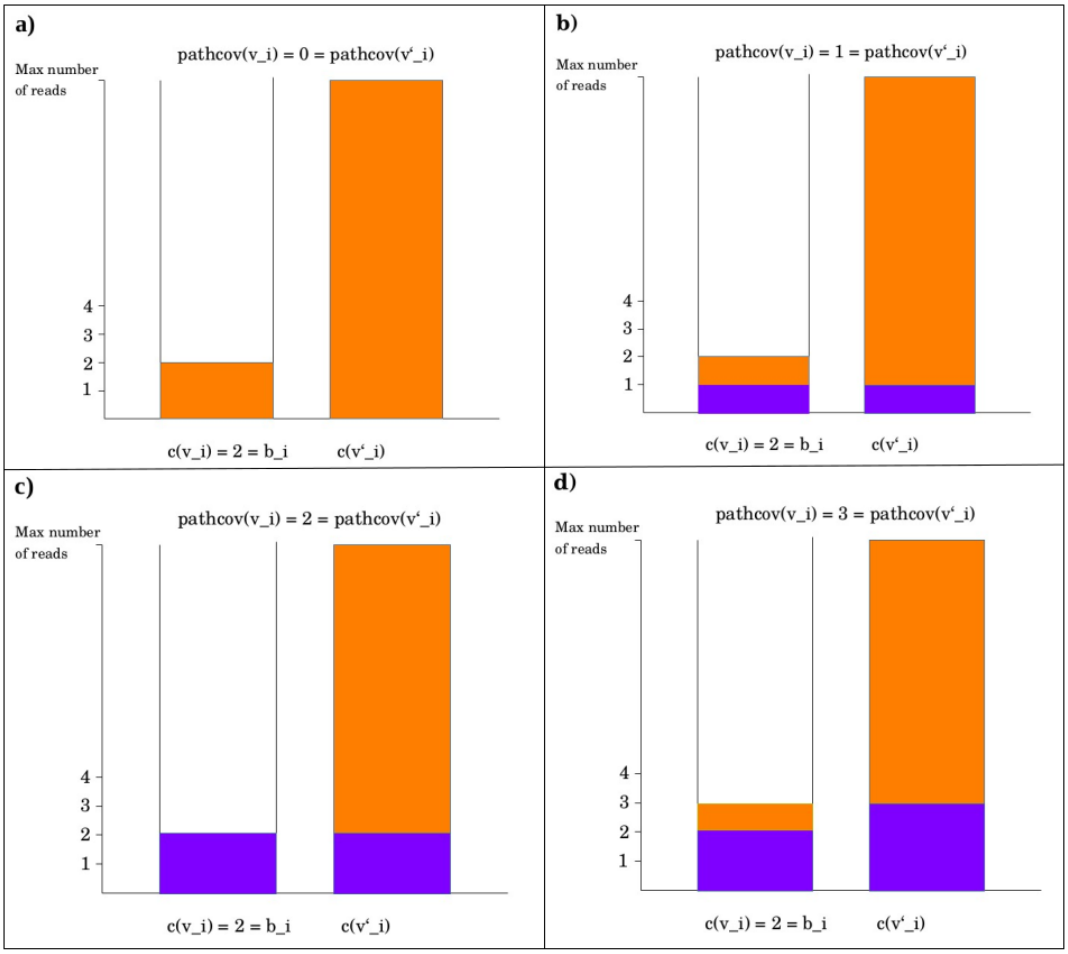

Figure 4: An example for two nodes $v_i, v_i' \in V$ from I', created from an element $i \in N$ with $b_i = 2$ in I. We set $c(v_i) = 2$, and $c(v_i')$ to a number that is larger than any possible coverage for a node. $pathcov(v)$ denotes the coverage for a node v. The orange areas in the diagrams denote the remaining penalties for the nodes. The blue areas mark the coverage for the nodes, as long as it is not redundant. If the coverage is greater than $c(v)$ for a given node v, the redundant coverage is penalized. Hence, orange area is added for that node. Now, we look what happens if read coverage increases. As long as coverage is below $b_i = c(v)$, the summed penalty for both nodes decreases when the coverage increases. In the example, this means that if we have a coverage of 1 (subfigure **b**), the penalty is lower than if we have a coverage of 0 (subfigure **a**). For a coverage of 2, it decreases further (subfigure **c**). If coverage is equal to b_i or higher, the summed penalty remains constant if we increase coverage further. Thus, the total penalty for a coverage of 3 (subfigure **d**) is the same as for a coverage of 2. This way, we can translate the demands from I into penalties in I' even though we punish redundant coverage in Repeat Selection.

2.5 Output comparison

For the purpose of examining our results, it is necessary to have a method for comparing outputs. This is especially salient since for CHM13, there exists a reconstruction of the rDNA repeat sites that has recently been conducted by the T2T Consortium [6]. Comparing the outputs of both reconstructions, and studying their differences, opens up the possibility to better understand the results our model produces. Yet, even for resolving rDNA material from other samples, for which no other reconstruction exists, a method for comparison is useful, because it allows to see how different parameter settings in our model affect its output.

Our goal is to compare whole sets of repeat copies, rather than single copies. This means we need a suitable metric that, given two sets of repeat copies S_1 and S_2, allows us to measure how similar each copy from one set is to each copy from the other. For this purpose, we use a model that employs edit distance and *minimum weight perfect matchings* on bipartite graphs. In the following, we are first going give the definitions for these concepts.

Let us recall the definition of a bipartite graph. The definition is cited after [27]. Since we took it from continuous text, we had to change the formulation slightly.

Definition 1. A *bipartite graph* is an undirected graph $G = (V, E)$, where V is partitioned into two sets, V_1 and V_2, such that there are no edges between vertices in V_1 and there are no edges between vertices in V_2.

What we want is a complete bipartite graph, which is a special kind of bipartite graph.

Definition 2. A bipartite graph $G = (V_1, V_2, E)$ is called a *complete bipartite graph*, if for all pairs of nodes v_i, v_j with $v_i \in V_1, v_j \in V_2$, there exists an edge $e = \{v_i, v_j\} \in E$.

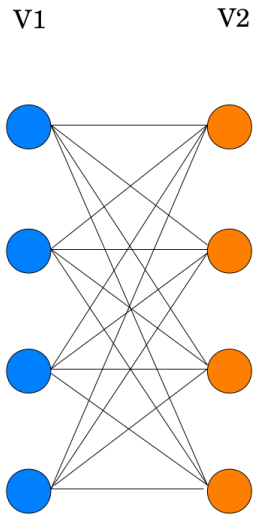

Figure 5: A complete bipartite graph with the two subsets V_1, V_2.

We say that the two sets of nodes V_1, V_2 correspond to S_1 and S_2. Each edge between two nodes $v_i \in V_1$, $v_j \in V_2$ has a label that denotes the similarity between the two corresponding strings. For expressing the distance between each pair of strings, we use the classic model of edit distance. We use the simplest variant of the model, the Levenshtein distance [28] or *unit cost edit distance* [27]. For abbreviation, we will simply refer to the unit cost edit distance as *edit distance* in the text.

Definition 3. Given two strings s_1, s_2 over an alphabet Σ, the *unit cost edit distance* $d(s_1, s_2)$ equals the minimum number of edit operations that are necessary to transform s_1 into s_2. As edit operations, we understand (1) Replacing a character in a string with another character from A, (2) Inserting a character into a string, and (3) Deleting a character from a string.

Since it is time-consuming to compute edit distances, we introduce a cutoff $c \in \mathbb{N}$ for the edit distance computation. When we compare two strings, and their edit distance surpasses c, we skip the computation. Instead, we label the edge between the two corresponding nodes with a high dummy value. Since we are primarily interested in strings that are similar to each other, we can include c into our model without losing valuable information. We will soon see how c allows us to increase the speed of computation. Now, we can define the problem we want to solve.

Problem 4.

COMPLETE BIPARTITE EDIT DISTANCE GRAPH

Input: Two sets of strings S_1, S_2.
A cutoff $c \in N$.

Output: A complete bipartite graph $G = (V_1, V_2, E)$ where the nodes in V_1, V_2 correspond to the sequences in S_1, S_2, and a function $f : E \to N$, such that

$$\forall\ e = \{v_i, v_j\} \in E:\ \ f(e) = \begin{cases} d(s_i, s_j) & \text{if} \quad d(s_i, s_j) \leq c \\ max\{|s_i|, |s_j|\} & \text{else} \end{cases}$$

The problem can be solved in polynomial time. For computing the edit distance between two strings, we can construct an optimal global alignment via Dynamic Programming with the Needleman-Wunsch Algorithm [22], where we set mismatch, insertion and deletion cost to 1, and match benefit to 0. This way, we can compute edit distances for all pairs of strings $s_i \in S_1, s_j \in S_2$ in $O(|S_1| \cdot |S_2| \cdot max\{|s_i|,\ s_i \in S_1\} \cdot max\{|s_j|,\ s_j \in S_2\})$ time. The term $max\{|s_i|,\ s_i \in S_1\}$ denotes the length of the longest string in S_1. The runtime results from the fact that we need $O(|s_i| \cdot |s_j|)$ time for computing an optimal global alignment for two strings s_i, s_j, while we need to compute optimal global alignments for all pairs of strings where one string is from S_1, and the other from S_2. Once we have the edit distance values, constructing the complete bipartite graph with the labeled edges is trivially possible in $O(|V| + |E|)$ time.

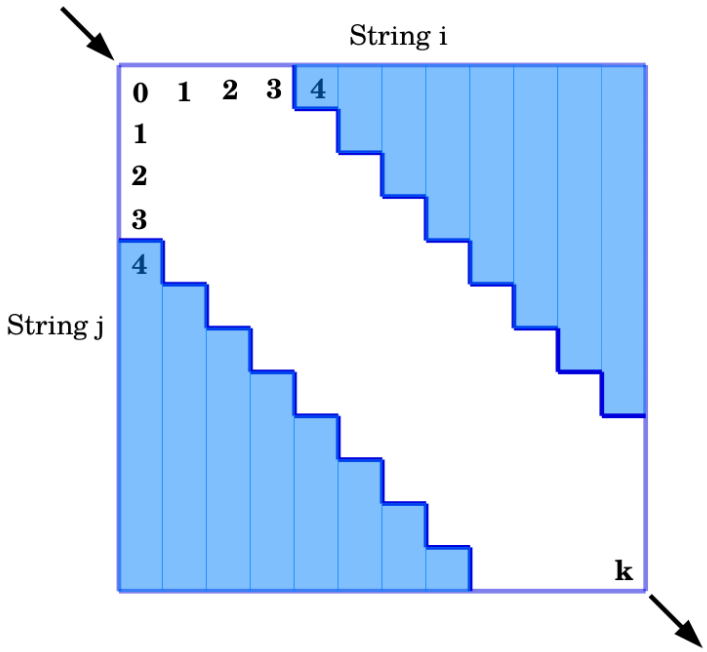

Figure 6: A simple form of banded DP for faster computation of an optimal global alignment for two strings. Here, we have chosen a cutoff of $c = 3$. We only compute fields with a distance of at most 3 from the diagonal, since all other entries can only be reached through more than 3 indel operations.

Since we have to deal with sets of hundreds of repeats, and every single repeat contains more than 40000 characters, alignments take a lot of time in practice. In order to speed up the computation, we thus use the technique of banded Dynamic Programming [23][24]. This means that we use our cutoff c and, for each row of the DP matrix, we only compute entries that have a distance of at most c from the diagonal. That way, we have to compute at most $2 \cdot c$ entries in each row. If, for example, $c = \frac{max\{|s_i|,|s_j|\}}{10}$, we only have to compute at most 20% of all entries in the DP matrix. See figure 6 for a visual example. In addition, if we have reached a row where no entry with a value smaller or equal to c exists, we can skip the computation. Note that for the purpose of this thesis, we only have implemented a naive and basic variant of banded DP, and a number of more sophisticated speedups are possible.

If we want to compare rDNA repeat copies, for example the ones from our model with the ones contained in T2T-CHM13, we have to keep in mind that the rDNA repeat sites are tandem arrays. This means they are cyclic, and it is hard to pin down exact start and end positions for the single repeat copies. Basically, one can choose any point, as long as the choice is made consistently over the whole repeat array under examination. However, if we compare our ouput with the sites in CHM13, it can be hard to find start and end points for the copies from CHM13 that match perfectly with the start and end of our copies. To account for this problem, we allow for free shift in our implementation of global alignment. This means, one does not have to enter and leave the DP matrix at the top left, respectively the bottom right corner. Instead, one can start at most m fields below, or at most m fields to the right of the

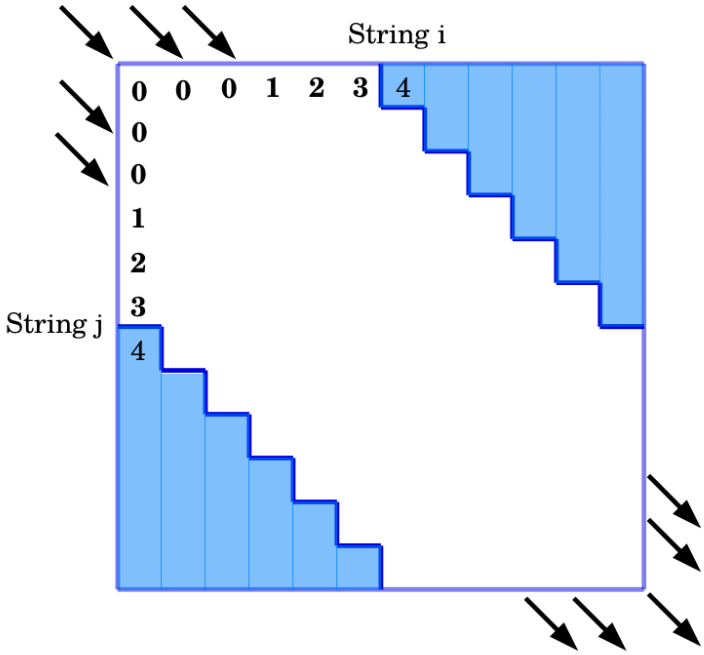

Figure 7: Banded DP with free shift. We have set $m = 2$, what means that one can start at most two fields away from the top left entry without extra costs. One can leave the DP matrix at the minimal entry from the five fields at the bottom of the diagonal. If we leave the cutoff c the same, and add a free shift value m, the size of the band is $2 \cdot (c + m)$.

top left entry, without costs. The same goes for leaving the DP matrix, where one can pick the minimum from the last m fields of the bottom row, and the bottom m fields of the rightmost column. The value m can be determined by the user. This way, we hope to ensure meaningful comparison of copies, even if we do not find start points that match well. For a visual example, see figure 7.

Once we have the complete bipartite edit distance graph for two sets of rDNA repeat copies, we have a number of possibilites for comparison. One is to compute a *minimum weight perfect matching* for the graph, that is, a subset of the graph edges such that each node has a degree of exactly 1, and the sum of the edge weights is minimal. This way, we can assess the overall similarity of the two sets, since we can see whether there are many pairs of similar copies even if each node from one set can only be connected to one node from the other set. Note that it is possible that the output sets are of different sizes. In this case, we can add dummy nodes and edges with high values to obtain sets of equal size, and to compute a valid perfect matching. Besides the minimum weight perfect matching, there is other valuable information we can obtain from the graph. For example, we can compare how many nodes from each of the two sets are connected to least one edge with a label that is below the cutoff c. If the number of such nodes is far greater in one set than in the other, this is an indicator for greater variation between the copies in the set where fewer nodes have good matches.

2.6 Output checking

In the absence of direct observations or a well-established ground truth, validation of complex new findings, such as hundreds of long sequences of DNA, is a difficult subject-matter. Validation strategies can easily become circular, since proof of a model's correctness would presuppose some degree of knowledge about the very object the model is supposed to newly reconstruct. In other words, evaluating output becomes a hard task if we assume little or no knowledge about how it *should* look like. This evokes the question how we can still be reasonably certain that our model produces good results.

To back up the claim that our model yields plausible output, we pick a unique subsequence of each repeat copy from the output and look whether we can find these sequences in the PacBio HiFi reads we use as input. To be precise, from each repeat copy, we want the shortest subsequence that occurs exactly once in the entire set of repeat copies. The idea is that, since the HiFi reads have an error rate of only 0.2% [7], a sequence that occurs in one of the *real* rDNA repeat copies should have exact matches in some of the reads, if it is not longer than a few hundred base pairs. Conversely, a sequence that does occur in the output of our model, but not in the real rDNA repeats, should be absent from the reads. This is because it is unlikely that errors in the reads, and errors in our model, produce exactly the same false sequence. Since each of the sequences we test for occurs in *exactly* one repeat copy from our output, we can check for each particular copy whether the variation it represents is plausible. Again, we should also note that most errors in the PacBio HiFi reads relate to homopolymer runs. Thus, we can increase the reliability of this method by applying homopolymer compression to both the output of our model, and the HiFi reads.

For our method to work, we need to extract suitable sequences from the repeat copies that form the output of our repeat selection model. The goal is to find subsequences for each copy that occur in none of the other copies, and that are as short as possible. A close relative is the problem of finding a shortest unique substring (SUS) for a given string. The method for solving our problem will build heavily on methods for finding shortest unique substrings. Let us first give the definition of a SUS.

Definition 4. Given strings s, s^*, where s^* is a substring of s, we call s^* a *shortest unique substring* of s if the following requirements are fulfilled: (1) s^* occurs exactly once in s, and (2) There exists no substring s^{**} of s with $|s^{**}| < |s^*|$ that occurs exactly once in s.

Based on the concept of shortest unique substrings, we define Problem 5: SHORTEST IDENTIFIERS.

Problem 5.

	SHORTEST IDENTIFIERS
Input:	A set of strings $S = \{s_0, s_1, ..., s_n\}$.
Output:	For each $s_i \in S$, the shortest substring s_i^* of s_i that (1) Occurs only once in s_i, and (2) Occurs in no other string in S.

For now, we call the substrings we want to find *identifiers* because each of them is unique to one string from the set, and hence identifies it. The problem of finding the shortest identifiers can be solved efficiently. For this, we use modifications of data structures that allow for the efficient search of shortest unique substrings, namely *suffix arrays* [29] and *LCP arrays* [29]. We give definitions of these structures borrowed from [27] due to their succinct character.

Definition 5. The *suffix array* $SA_S[1..n]$ of a text $S = s_1 s_2 \cdots s_n$ is the permutation of $[1..n]$ such that $SA_S[i] = j$ iff suffix $S_{j..n}$ has position i in the list of all suffixes of S taken in lexicographic order.

Note that suffix arrays can be built in $O(n^2 \cdot \log(n))$ with a naive approach, while it is also possible to build them in linear time using induced sorting [30]. The suffix array can easily be modified such that it contains the lexicographic ordering of all suffixes from a set of strings, instead of just one string. This structure is called a *generalized* suffix array [31]. It is also possible to invert suffix arrays. While a suffix array contains the starting positions of all suffixes of a string, sorted in lexicographic order, the *inverted suffix array* contains all lexicographic ranks of the suffixes, sorted in the order of the suffixes' starting positions. Regarding the order of the starting positions for suffixes from a set of strings $S = \{s_0, s_1, ..., s_{m-1}\}$, we say the first $n_0 = |s_0\$|$ positions are occupied by the suffixes from s_0, the next $n_1 = |s_1\$|$ positions are occupied by the suffixes from $s_1\$$, and so on. You can thus find all lexicographical ranks for suffixes from s_0 in the generalized suffix array by querying $1, 2, ..., n_0$ in the inverted suffix array. The same goes for all other strings and their respective intervals in the inverted suffix array.

Another important data structure for finding the identifiers, that is closely related to the suffix array, is the LCP array:

Definition 6. Given a string $S = s_1 s_2 \cdots s_n$, with $s_n = \$$, and a suffix array $SA_S[1..n]$, a *longest common prefix array* $LCP[2..n]$ is an array such that $LCP[i]$ stores the length of the longest prefix that is common to suffixes $S[SA[i-1]..n]$ and $S[SA[i]..n]$.

As the definition was taken from continuous text, we made some slight changes in the formulation. For each suffix, the LCP array stores the length of the longest prefix it has in common with its lexicographical predecessor. LCP arrays can be constructed in $O(n)$ time using Kasai's Algorithm [32]. For fast search of SUS, there are different ways [33]. One is to use the LCP

array. For each suffix s', we can compare its LCP value with the LCP value of its lexicographical successor. That means, we compare the longest common prefix of s' and its lexicographical predecessor with the longest common prefix of s' and its lexicographical successor. The maximum of both values denotes the length of the longest non-unique prefix of s'. This works correctly because the suffix with which s' shares the longest prefix must be one of it its lexicographical neighbors. To get the length of the shortest unique prefix of s', we can just add 1 to the length of its longest non-unique prefix. If we query the shortest unique prefixes for all suffixes of a string and pick the shortest one among them, we know the SUS. This works because, in a string, each substring is the prefix of a suffix of the string. Thus, the shortest unique prefix is also the SUS. The only exception is when the longest non-unique prefix of a suffix is the suffix itself. This time, we cannot add 1 to get a non-unique suffix, since we have reached the end of the string. We thus have to catch this special case to get a correct result. Since we know the starting positions of all suffixes, we can also infer start and end position of the SUS.

If we have a set of strings S, and want to find the shortest identifier s_i^* for a string $s_i \in S$, we can proceed analogously. First, we build a generalized suffix array that stores the lexicographical ranks of all suffixes for the strings from S. We also build the corresponding LCP array, and the inverted generalized suffix array. Then, we query the inverted generalized suffix array for the lexicographic ranks of all suffixes from s_i. We use the LCP array to identify the shortest unique prefixes for all these suffixes. Since we compare the suffixes with the closest lexicographical neighbors from *all* strings, we always get the shortest prefix that is unique in the entire set of strings. From the shortest unique prefixes of all the suffixes, we can then again pick the shortest one. This way, we get the shortest substring from s_i that is unique in the entire set of strings. Hence, we have our identifier s_i^*.

Once we have the set of identifiers, we can search them in the HiFi reads. As mentioned above, we can apply homopolymer compression to both the reads and the output from our repeat selection model to reduce error rates. This means, we compute the shortest identifiers in the compressed repeat copies, and search them in the compressed HiFi reads. The low error rate of the compressed reads allows for exact pattern matching. As long as we can find identifiers with a length of, roughly, 1000 bp or less, we can confidently search for them in the reads without having to fear a large number of false negatives. If an identifier is not in the read data, this is likely due to an error by our repeat selection model. Conversely, if an identifier *is* present in the read data, we do not expect it to be a false positive, unless the identifier is very short. This is because it is unlikely that an error in our model produces by chance a sequence that actually is in the data. Moreover, we count the number of reads in which an identifier is found. So, we can see whether a sequence occurs very often in the reads, what suggests that it is present in more copies than our model predicts, or very rarely, what might indicate a false positive.

3 Results

With the repeat selection model, we assembled the rDNA repeat copies from CHM13, as well as from five Human Pangenome Reference Consortium (HPRC) samples. As described above, an assembly by the T2T Consortium already exists for CHM13, allowing for comparison of the results. In the following, we first describe our assembly of the CHM13 repeats. On the one hand, we look at how well the repeat selection model performs on the input graph. For this, we examine how much of the coverage on nodes with different length and coverage depth in the assembly graph the model is able to explain. On the other hand, we look at the reconstructed repeat copies themselves, and present an IGV visualisation of an alignment of the copies against the canonical rDNA unit. Based on the methods described in sections 2.5 and 2.6, we then move on to compare our reconstructed repeats for CHM13 with the T2T assembly, and analyse them further. Finally, we present our reconstruction of the repeat copies from the five HPRC samples.
The source code for the repeat selection model, as well as for the output comparison and output checking methods, can be found on GitLab[3].

3.1 Reconstruction of rDNA repeats from CHM13

3.1.1 Preprocessing and parameter settings

For our assembly of the CHM13 repeats, we used PacBio HiFi and ONT alignments corresponding to version 1.0 of T2T-CHM13. The data is available online[4]. For version 1.0 of T2T-CHM13, the reads were already aligned to the rDNA sites, but not resolved for single repeat copies. This means the alignments contain pileups of reads on the sites were the rDNA regions are located on the respective chromosomes. We thus did not have to search the reads that match to the rDNA sites ourselves, as we would later have to do for the HPRC samples.

From the HiFi reads from all the five sites, we constructed a blunted assembly graph using MBG. As k-mer size, we used 351. Regarding the k-mer and unitig abundance cutoffs, we choose a value of 10 after experimenting with a number of different cutoffs. The resulting output graph consisted of 4237 nodes and 5206 edges. The largest connected component made up 65.14% of the graph, while 6.34% of the nodes were isolated. Average coverage depth on the nodes was 27.6. The largest connected component showed a cyclic structure, as expected when assembling a graph from highly similar tandem repeats. Figure 8 is a Bandage [15] visualization of the assembly graph, focusing on the largest connected component.

[3] *https://gitlab.cs.uni-duesseldorf.de/albi/albi-students/ma-frederik-oehl*

[4] *https://github.com/marbl/CHM13#telomere-to-telomere-consortium*

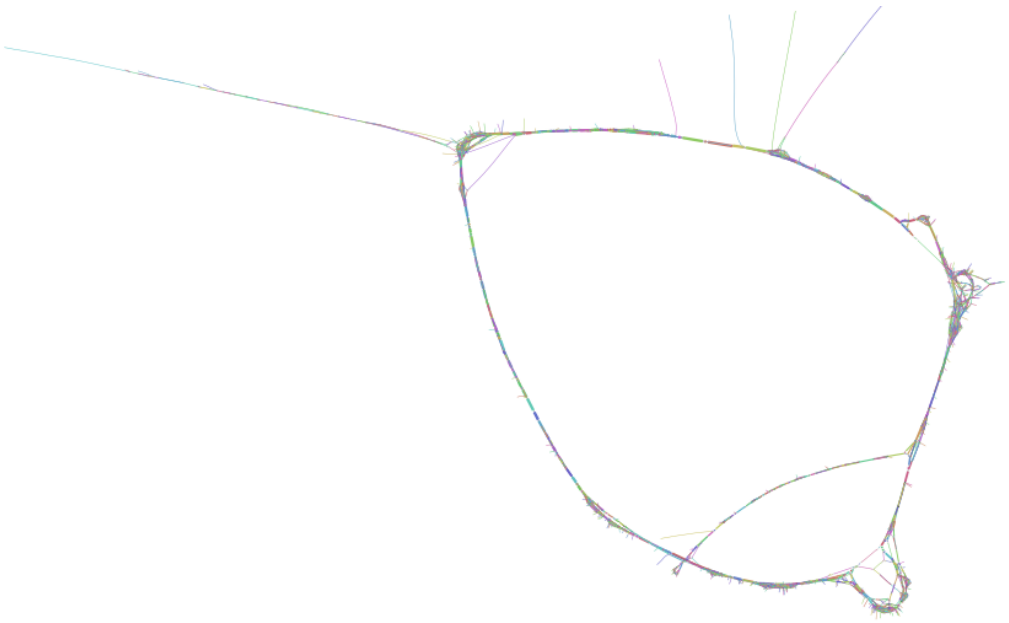

Figure 8: A Bandage visualization of the blunted assembly graph for CHM13. This image focuses on the largest connected component, with its characteristic cyclic structure.

As described in section 2.2, the next step consisted in breaking the cyclic structure. For this, we first aligned the canonical rDNA unit KY962518.1 to the graph using GraphAligner. As expected, the alignments roughly followed the circular struture of the largest connected component, allowing us to break the circle in a position that corresponds to the beginning and end of the canonical rDNA unit. For breaking the graph's cyclic structure, we removed 12 edges in total. Figure 9 shows a Bandage visualization of the graph, after breaking the cycle.

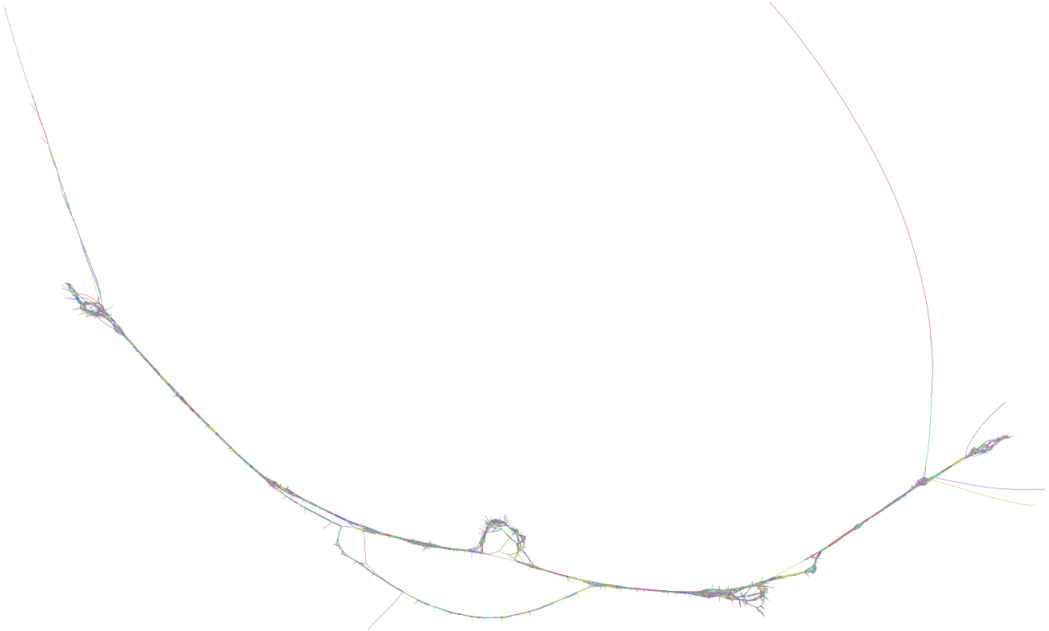

Figure 9: Bandage visualization for the CHM13 assembly graph, after the removal of edges to break the graph's cyclic structure.

The next step consisted in aligning the ONT reads to the graph. For this, we used GraphAligner with default settings, meaning that alignments are considered valid if they have a similarity of at least 65% to the corresponding sequence of base pairs formed by the graph nodes. From the resulting, very large set of alignments we kept only the best one for each ONT read. From these alignments, in turn, we kept only those that are at most 5% longer or shorter than the canonical rDNA unit, meaning those that have a length between 42.5 and 47 kbp. This restriction means that we might miss more exotic copies with massive insertions or deletions, as they can sometimes occur [5]. But since we do not know how often copies with strongly divergent lengths are present in the DNA, and want to avoid a scenario were the model selects copies that are useful for the optimization, yet much too short or too long to be realistic, we decided to make this restriction. It is, of course, also possible to set different cutoffs. Later, in the HPRC samples, we choose a more relaxed threshold of $+/-$ 10% of the canonical rDNA unit's length. For CHM13, we found a total of 1517 alignments among the best alignments for the ONT reads, that have the desired length. These form the universe from which we select the optimal paths.

After the alignment, the next step was to determine the values for the number of paths to select k, the coverage c_{avg} that a single selected path explains on a node, and the weight w for the nodes. We derived k from counting the number of HiFi reads both on the entire genome, and on the rDNA repeat sites. Since we know the length of the human genome, and have plausible estimates for the average length of a single rDNA repeat copy, we can thus calculate the estimated number of repeat copies in our sample. This way, we arrived at $k = 221$, which is very close to the T2T Consortium's estimate of 219 copies [6]. For c_{avg}, we started with the average node coverage given by MBG (27.6). By running the model with different values for c_{avg} and using binary search, we found out that the value of the optimization function becomes minimal for $c_{avg} = 17.5$. We decided to work with this experimentally determined value. Regarding the weight w, the main goal we want to achieve by introducing this value is to include the length of the sequences corresponding to the graph nodes into the calculation. Since longer nodes account for a greater part of the selected copies' sequence, we want the model to pick those with a higher priority. We decided to set $w(v) = \log(|sequence(v)|)$ for each node v. We see this as a compromise between not prioritizing length at all ($w(v) = 1$ for all v), and possibly overprioritizing it by setting $w(v) = |sequence(v)|$, where some nodes would have hundreds of times more weight than others.

3.1.2 ILP performance

As input for our model, we used the entire graph except for the isolated nodes. Furthermore, we used those ONT alignments that fulfill the requirements, and the parameter settings described above. We decided to remove the isolated nodes, since we considered it very unlikely to find a single node that would serve as an entire repeat copy. We kept all the other compo-

nents of the graph, though using only the largest component might also be a possibility worth considering. We formulated our model as an Integer Linear Program (ILP) and ran it on the free solver that comes with Python's PULP package (COIN) and, later, on the commercial solver Gurobi 9.1.2. With COIN, we could reduce the gap between lower bound and best found solution to under 0.5% within less than 300 seconds. Reaching an optimal solution, however, turned out to be infeasible: even after 750000 seconds (208.3 hours), there was still a gap of 0.36%. In a later 250000 seconds (69.4 hours) run with Gurobi, we found a solution that is about 0.1% better than the previous one after about 4400 seconds, but still could not prove its optimality. A gap of 0.03% was still left over in this hitherto longest Gurobi run. Note that the results discussed in the following come from the long COIN run, since the Gurobi run was only performed at a later stage of the thesis, when the time window was narrower. Based on the best bound found with COIN, this solution is a 1.0036-approximation of the optimal solution. If we take the best bound we later found with Gurobi into account, we can even say that it is a 1.0015-approximation. Thus, we consider it meaningful enough for the analysis. For the later experiments with the HPRC samples, we used Gurobi in all five cases. Note that on four out of five HPRC samples, we could solve the REPEAT SELECTION problem to optimality without problems, what indicates that this is possible for most instances, while the CHM13 instance appears to be an unlucky case.

Regarding the output, we have 2739451.9 worth of weighted coverage that is either under- or overexplained, with the best bound we found as yet being 2735590.9. The total, weighted coverage of all nodes before running the model is 7800128.3. If we only count nodes that are covered by ONT reads, the values are 2661516.9 and 7722193.3. Thus, the model explains about 65% of the coverage on the nodes. The top row in figure 10 shows how strongly the model over- or underexplains the coverage of the nodes on the input graph. For many nodes, the value is close to 0, meaning there is little over- or underexplanation of their coverage. Yet, there are clear exceptions. For one, we have about 700 unexplained nodes, meaning their coverage is the same as before the run. All of these are nodes that are not covered by any ONT alignment, so, by definition, the model cannot explain them. Also, as we explain below, almost all have very low coverage. Perhaps more important are nodes whose coverage is overexplained by more than 100%, what means they have more coverage after the run, than before. In the most extreme case, a node is overexplained by a factor of about 250. There is one more case of overexplanation by a factor of over 100, and four cases of overexplanation by a factor of 40 to 50. For the plots in figure 10, we have removed these outliers and set a cutoff of -10 and -20, respectively. The plots in the middle row of figure 10 show the over- and underexplanation for all nodes, versus their coverages before the run, or $c(v)$ values. If we take the $c(v)$ value into consideration, we see that it is very low for almost all of the strongly overloaded nodes. Generally, the more coverage a node has before the run, the better it is explained, though there is some bias towards underexplanation on nodes with very high coverage values. For the unexplained nodes, it is true as well that they tend to have very

low $c(v)$ values. In the left plot in the middle row, the points that correspond to unexplained nodes are located to the very right. Almost all are concentrated in the bottom right corner, and are visible as one green dot. When we plot over- and underexplanation of nodes against the lengths of their corresponding sequences (bottom row in figure 10), we observe a similar trend: if a node sequence is longer, it is more likely that the model explains its corresponding node well. The general tendency seems to be that the model overloads short nodes with low coverages in order to explain 'pricey' nodes, though other factors are important as well. Naturally, it is easier to overexplain a node with low coverage, as this requires only a few paths. The results from section 3.1.3 suggest that coverage fluctuation on the graph is another contributing factor.

3.1.3 Visualization of unexplained coverage on the graph

In order to find out where the overloaded and unexplained nodes are located, we created another Bandage visualization of the blunted assembly graph, and color-coded its nodes based on how strongly the model over- or underexplains their coverage. We color unexplained nodes in plain red (X11 color: #ff0000). For nodes that are underexplained by less than 100%, we use five different shades of red, depending on how close the model comes to a perfect explanation of their coverage. To be more precise: If a value of 0 means that a node is perfectly explained, while a value of 1 means that it is unexplained, or underexplained by 100%, we divide the interval $(1, 0]$ into sub-intervals $(1, 0.8]$, $(0.8, 0.6]$, and so on. Nodes on which the model explains at most 20% of coverage fall into the interval $(1, 0.8]$, and we assign the X11 color #ffafaf to them. Nodes on which the model explains more than 20%, but no more than 40%, fall into the interval $(0.8, 0.6]$, and we assign the X11 color #ffbfbf to them. Thus, the more coverage the model explains on a node, the lighter is the shade of red we assign to the node. For overexplained nodes, we do the same thing in reverse, but use shades of blue instead of red. If the model overexplains a node by less than 20%, we assign the X11 color #efefff to it. If the model overexplains a node by at least 20%, but less than 40%, we color the node in #dfdfff. We use plain blue (#0000ff) if a node is overexplained by at least 100%, but less than 120%. If a node is overexplained by more than 120%, we use darker tones of blue. For outlier nodes whose coverage is explained more than 50 times, we use the color black.

Figure 11 shows the largest connected component of the graph. Note that the thickness of the nodes in the figure is directly proportional to their pre-run coverage $c(v)$. If a node u has a coverage of 30 before we run the model, while a node v has a coverage of 300, v will be ten times thicker than u in the figure. This way, it becomes easier to see how much of the total coverage on the graph the model explains. The figure allows us to make some interesting observations. Especially, one can see that most thick nodes in the bottom-left part of the graph are slightly underexplained, whereas, in the top-right part of the graph, they are slightly overexplained. At the top-right end, slight underexplanation becomes the norm again. This suggests that the

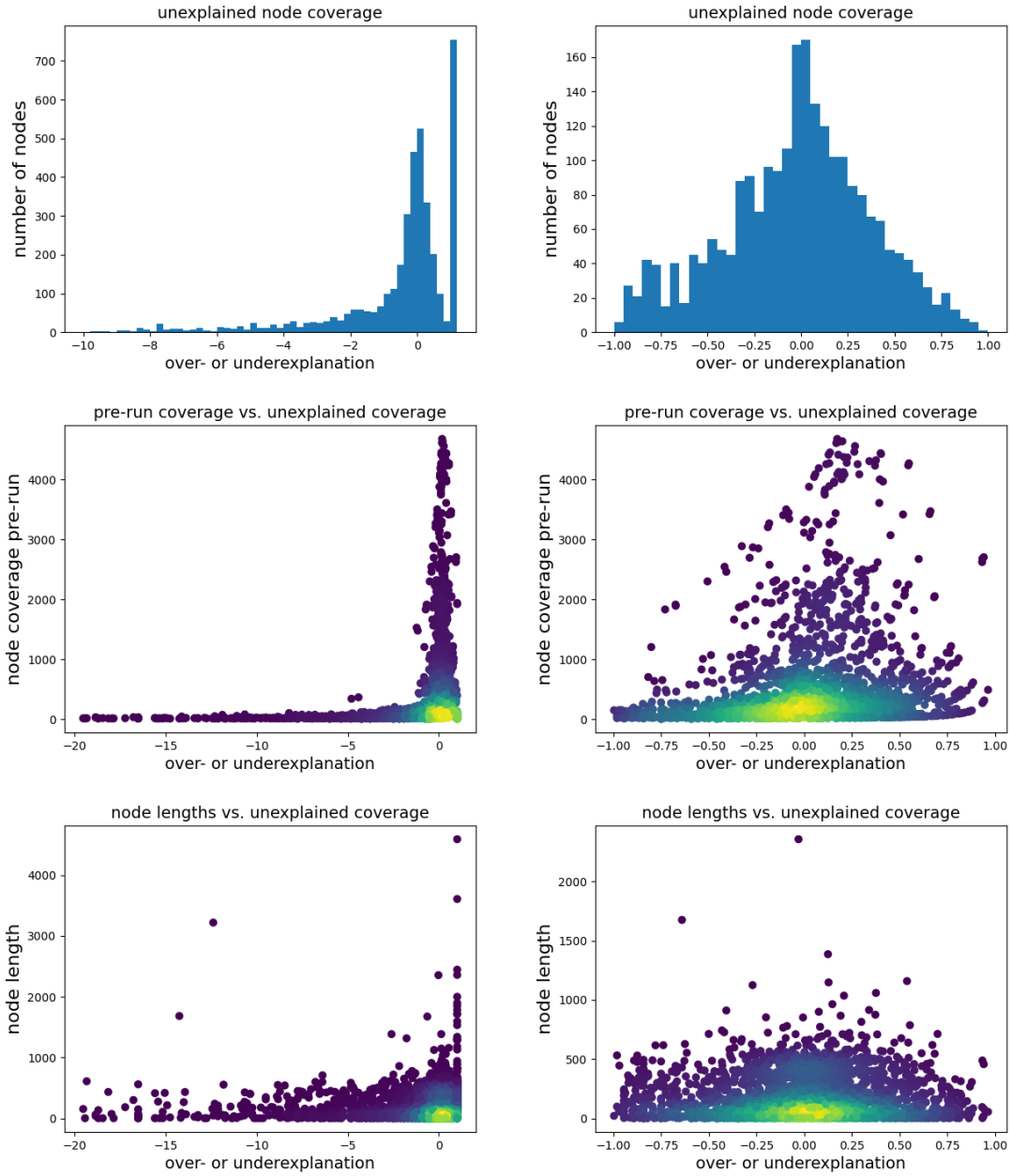

Figure 10: Explanation of coverage on the nodes of the CHM13 graph. In the histograms in the top row, the x-axis denotes how strongly the coverage is under- or overexplained. A value of 1 means that a node is not covered by any path. A value of 0 means the coverage on a node is perfectly explained. A value of less than 0 means that a node is covered by more paths than necessary to explain its coverage. The y-axis shows how many nodes fall into each interval of size 0.2 (left) or 0.05 (right) on the x-axis. The right histogram is identical to the left one, but zoomed in such that only values in the interval $(-1, 1)$ are shown.
In the scatter plots in the middle row, the x-axis is the same as in the top row. The y-axis now shows the coverage depths for the nodes prior to the application of the model, also referred to as $c(v)$ values. The color of the points shows the density of the plot. Where the points are blue, there is low density, meaning fewer datapoints fall into the given area. Yellow color denotes high density. In the scatterplots in the bottom row, under- or overexplanation is plotted against node lengths.

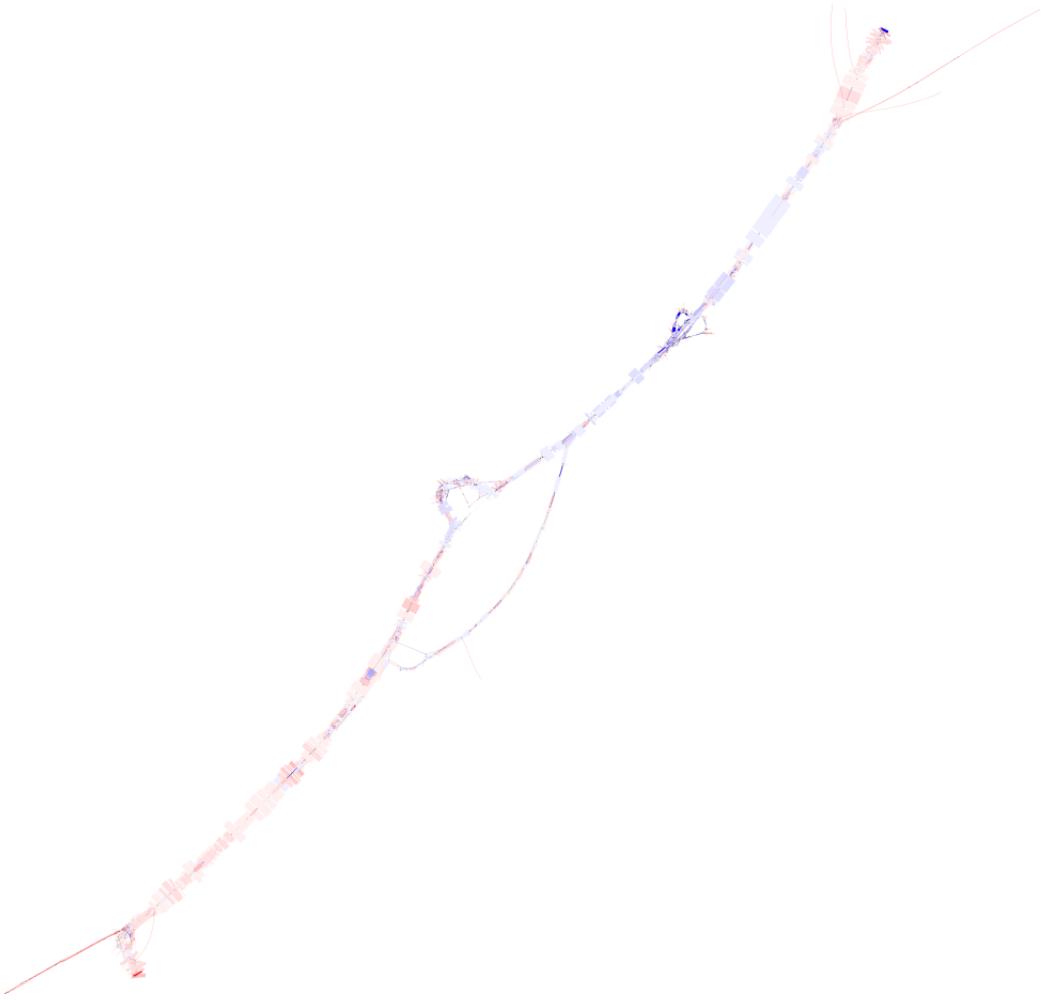

Figure 11: The largest connected component of the blunted assembly graph. We color-coded the nodes, such that red color denotes that a node is underexplained, while blue color means that a node is overexplained. The darker the color on a node is, the stronger it is under- or overexplained. The thickness of the nodes is directly proportional to their $c(v)$ values.

HiFi read coverage for different regions of the rDNA repeat copies may fluctuate.

In order to see where the strongly overloaded nodes, as well as the unexplained nodes, are located, we need to zoom in. Figure 12 shows an excerpt from the top-right part of the graph, that contains a cluster of strongly overloaded nodes. We can see that the 'blue shift' in the nodes coincides with a chaotic region on the graph. Compared to neighboring regions, it contains more sub-paths of different lengths. In other, similar regions, we can observe similar blue shifts, though they are not as pronounced. In figure 12, even some of the thicker nodes are strongly overloaded. As the great majority of overloaded nodes in other regions of the graph have low $c(v)$ values, this is an unusual phenomenon. A possible explanation for blue shifts in chaotic regions are coverage drops in these regions. When the summed $c(v)$ value of all the nodes that are covered by potential paths becomes too small in a given region, the model is forced to overload some nodes. To answer the question why chaotic regions on the graph seem

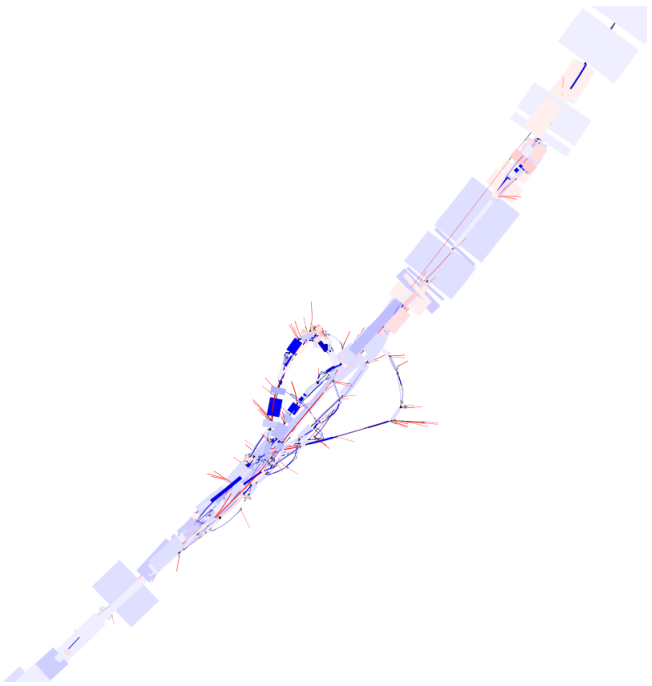

Figure 12: A cluster of strongly overloaded nodes.

to coincide with coverage drops, further study is necessary. Possibly, there are some regions on the rDNA repeats where the HiFi reads have unusually high error rates, what would lead to a high number of unique k-mers and unitigs for these regions.

Figure 13 shows the bottom-left end of the graph, where a number of unexplained nodes are located. In the left subfigure, unexplained nodes are colored in red, as described above. In the right subfigure, nodes that are not covered by any path from R_{Aln} are colored in white. By definition, the model cannot explain these nodes, since no ONT alignment contains them. As explained in section 3.1.2, none of the unexplained nodes on the CHM13 assembly graph is contained in any potential path. If a node is contained in one or more paths from R_{Aln}, it is also contained in at least one path from R_{Opt}. As a consequence, all of the unexplained nodes are invisible in the right subfigure.
Among the unexplained nodes in the figure, there are two with a high $c(v)$ value of around 1900. They are located at the bottom of the figure. These nodes may represent discrepancies between the HiFi and ONT reads, as their corresponding sequences are frequent in the HiFi-based assembly, but do not occur in the ONT reads. A far larger fraction of unexplained nodes are low-coverage nodes with only one neighbor on the graph. In the figure, they stretch out orthogonally from the main structure of the graph, and are thus visible as 'antennas'. Antenna nodes occur on the whole graph, but are most frequent in chaotic regions. This, as well as their large number and low coverage, makes it unlikely that they represent potential start or end points of repeat copies. Rather, we suggest to regard them as similar to isolated nodes.

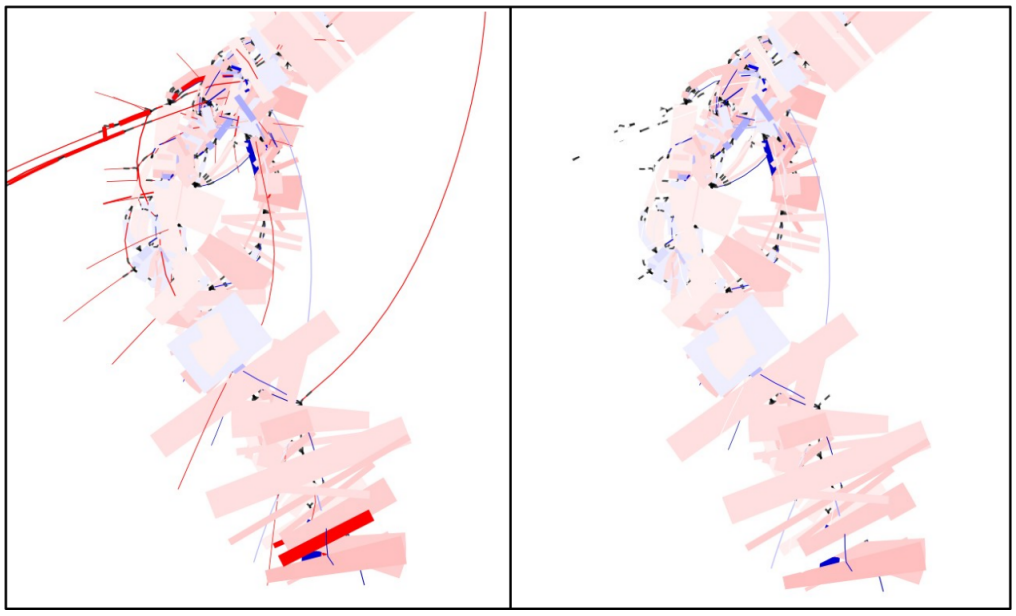

Figure 13: A zoom-in on the bottom-left end of the graph. In the left subfigure, we depict the nodes in the same way as in figures 11 and 12. In the right subfigure, we made nodes invisible with white color if they are not contained in any path in R_{Aln}, and hence cannot be selected by the model. The short, black lines in the right subfigure are edges between invisible nodes.

3.1.4 Repeat copies

As output, the repeat selection model yields the k individual rDNA repeat copies, where $k = 221$ for the CHM13 sample. For exploring the structure of the copies, we mapped all of them against the canonical rDNA unit KY962518.1, using minimap2 [16]. We found that many of the copies differ considerably from the canonical rDNA unit. Especially, they can contain larger insertions and deletions. When experimenting with minimap2 in order to produce the clearest alignment, we found that it works best to use the '-ax splice' command. The 'splice' command is normally used for mapping mRNA, where the non-coding regions have been removed, against DNA. This means that the tolerance for deletions in the alignment is increased. For aligning our rDNA copies, of which some contain indels with a length of 1000bp or more comapared to the reference unit, this approach yielded the cleanest results. In order to produce an instructive visualization of the alignment, as shown in figure 14, we used the Integrative Genomics Viewer (IGV) [17].

When we consider the visualized alignment, we can see that the coding region, which is located at 0 to 13 kbp on the copies, is altogether more stable than the non-coding region. In the coding region, the copies show polymorphisms with regard to the reference unit, but there are no massive insertions or deletions. Also, there is no large-scale variation between the copies in this region. Some parts of the coding region contain virtually no polymorphisms, while others are more active. The active regions are in rougly the same spots for all of the copies. When we look at the non-coding region, or intergenic spacer (IGS), the differences become

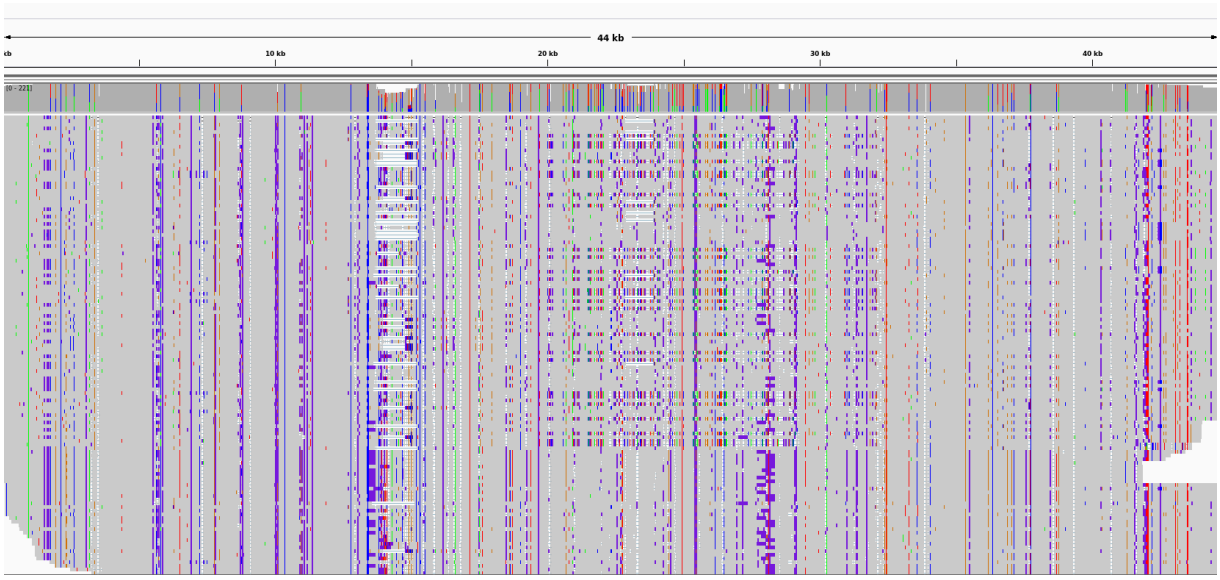

Figure 14: The CHM13 repeat copies that our model reconstructed, mapped against the canonical rDNA unit. The copies are represented by the pileup of sequences that makes up the large bottom part of the figure. Each horizontal segment in this pileup is one copy. Areas that are colored in grey show no difference to the canonical rDNA unit. White spaces denote deletions, while longer blue spaces denote insertions. Colored dots denote SNPs. We used minimap2 for the alignment, and IGV for the visualization. Approximately the leftmost third of the image represents the coding region of the copies, while the other two thirds correspond to the non-coding region.

more pronounced. For one, in the region that separates the coding region from the IGS, about a third of all the repeat copies show massive deletions with a length of 1000bp or more. Some also have large-scale insertions of about 800bp length. In the visualization in figure 14, deletions are indicated by white spaces, while insertions are indicated by longer stretches marked in blue. In the central part of the non-coding region, located roughly between 20 and 32 kbp, a fraction of copies shares a large number of point mutations and smaller insertions. The other copies are more similar to the reference unit in this region, except for 21 copies that have massive deletions at about 24 kbp. In contrast to some of the other parts of the IGS, the region between 32 kbp and 42 kbp appears to be very stable. Like in the coding region, the copies generally show little difference to each other, or to the reference unit. The differences to the reference unit tend to be even smaller than in the coding region, which does have some instability hotspots. Towards the end of the repeat copies, at around 42 kbp, there is another region that shows marked differences between the output copies and the reference unit, and between the copies themselves. These difference appear in the form of point mutations, as well as insertions. It is also noteworthy that for some of the output copies, the alignment does not encompass the first, or last, few thousand base pairs of the reference unit. It is not clear whether this is an error produced by our model, or whether some rDNA copies actually start later or end earlier, or whether they may differ greatly from the reference unit in the start and end regions.

3.2 Comparison to the T2T Consortium's Reconstruction

3.2.1 Repeat copies from T2T-CHM13

For comparing our output with the rDNA repeats from the T2T Consortium's reconstruction, we first extracted the individual rDNA copies from the T2T-CHM13 reference genome. For this, we aligned the last 100 bp of the canonical rDNA unit against the reference, in order to find plausible ending positions for each repeat copy. We treated each slice of DNA between two ending points as one individual repeat copy. For the copies at the beginning of each repeat region, we additionally had to search for a starting position with the first 100bp of the canonical rDNA unit. We extracted all 219 rDNA copies from T2T-CHM13. Note that there are only 32 different copies in the reference genome, as pointed out in section 1.2.1. In contrast, all 221 output copies from our model are unique, even though the model does allow for multiple identical copies. We visualized the T2T copies in the same way as we did with the output of our model. Figure 15 shows this visualization. Note that the T2T copies, as they are shown in the figure, are sorted according to chromosome of origin, so the order is not comparable to the order of output copies from our model above. Still, certain structural similarities are visible. Like in the output from the repeat selection model, the coding regions of the copies appear similar to one another, though there are a number of differences to the canonical rDNA unit. Between coding region and IGS, many copies show large deletions, or longer insertions. In the region between 20 and 32 kbp, some copies have a large number of polymorphisms. Between 32 and 42 kbp, the copies appear stable, while at the end of the copies, there is another region crowded with polymorphisms. These features are similar to the output from our model. Yet, the differences between the two reconstructions are significant, as further analysis shows.

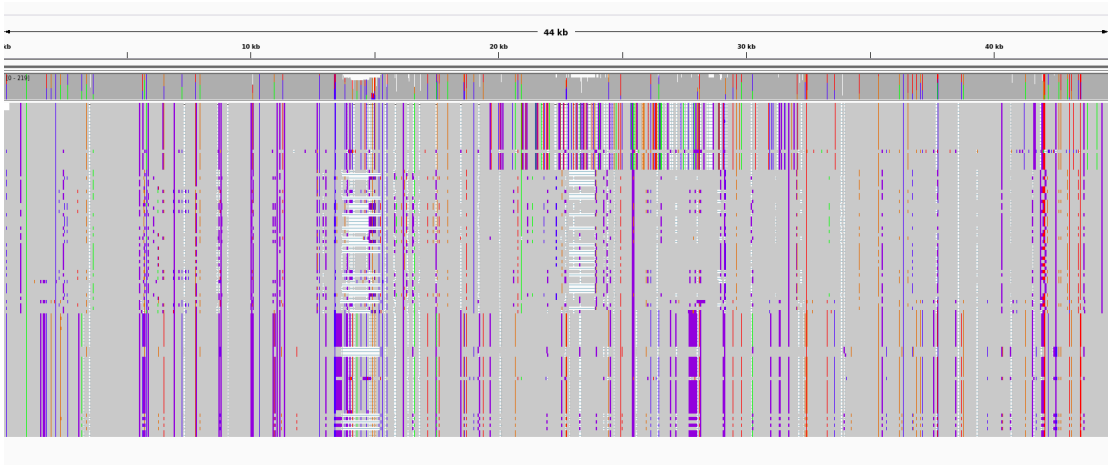

Figure 15: The CHM13 repeat copies from the T2T reference genome. We mapped the copies against the canonical rDNA unit, just like the output copies from the repeat selection model above. They are sorted according to their chromosome of origin, with the topmost ones from chromosome 13, and so on.

3.2.2 Edit distance-based comparison

For a copy-to-copy comparison of the two datasets, we computed a complete bipartite graph of the kind we described in section 2.5. The two sets of rDNA repeat copies function as the two sets of strings V_1, V_2. The edges in the graph denote the edit distances between each pair of copies. As cutoff c, we choose 4500, meaning that we label an edge between two nodes with the edit distance if the two copies have a similarity of, approximately, 90% or more. We allow for a free shift of up to 1000, in order to anticipate possible differences in the exact start points of the copies from the different sets. If c is surpassed, we set the edge label to the total length of the longer one of the two strings. This large dummy value ensures that, in the MWPM, edges between nodes with more than 90% similarity are selected with priority.
Strikingly, in the complete bipartite edit distance graph, only 54 out of 221 copies from the repeat selection model have an edge to at least one of the T2T copies, whose value is smaller than 4500. Conversely, 215 out of the 219 T2T copies are adjoined to such an edge. The MWFM contains only 53 pairs of nodes that are connected with an edge whose value is below 4500. Figure 16 shows a Graphviz-based [34] visualization of the bipartite edit distance graph, containing only the edges with values below 4500, and only the nodes that are adjoined to at least one such edge.

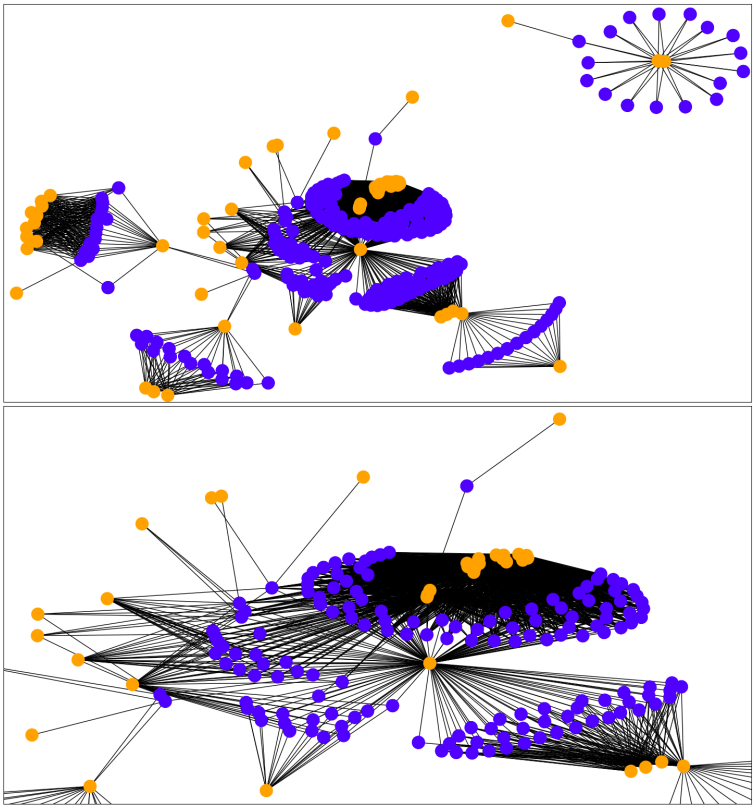

Figure 16: A visualization of the bipartite graph between the repeat copies from the two sites. All edges with values over 4500 were deleted, so the connections between similar nodes become visible. All nodes that became isolated through the edge removal were deleted as well. **Orange** nodes stand for copies from the repeat selection model, while **blue** nodes stand for T2T copies. The bottom picture is a zoomed-in excerpt from the top picture, that shows the top picture's center. The pictures were created with Graphviz [34], via a Python interface.

In the graph, **blue** nodes correspond to copies from the T2T reference genome, while **orange** nodes correspond to copies from the repeat selection model. As we can see, small numbers of orange nodes are often connected to large numbers of blue nodes. The most likely explanation is that clusters of blue nodes that are connected to the same few orange nodes represent identical, or very similar, repeat copies. After all, the T2T reference genome contains only 32 different copies, and the most abundant of these occurs 66 times. There are also some instances where clusters of orange nodes and clusters of blue nodes are densely connected, as on the left and bottom-left side of the top figure. In these cases, we presumably have a high similarity between a number of copies from the repeat selection model and a number of copies from T2T, as well as between copies that come from the same set. In other instances, orange nodes are only connected to a single blue node. Thus, some of the copies from our model that do have similar 'partners' in the T2T stack, do not seem to have much in common with any of those copies that are abundant in the DNA according to the T2T reference genome.

On the whole, a smaller subset of rDNA copies from our model is similar to a far larger subset of the T2T copies. For many copies from our repeat selection model, there exists no similar copy in the T2T reference, while for almost all of the copies from T2T, there exists a similar copy from our model. Given figures 14 and 15, such a big difference was not necessarily expected, since, even though the two visualizations may not look similar in all regards, they do at least show some structural similarities. It is possible that the IGV-based visualization overemphasizes similarities between the two datasets, though further analysis is required to find out how and why this might happen.

3.2.3 Analysis and comparison through shortest identifiers

We interpret the results from the edit-distance based comparison as evidence that, compared to the T2T reference genome, the repeat selection model predicts significantly higher variation in the rDNA repeats. With the method from 2.6, we investigated this finding further, in order to find out whether the prediction that our model makes is justified. For each of the different repeat copies from both datasets, we determined the shortest identifier, in order to search it in the set of HiFi reads from CHM13. If the shortest identifier from a given repeat copy is present in the read data, we regard this as evidence that the copy, or at least those features of the copy that tell it apart from all other copies, is present in the real DNA.
We conducted two runs: one with homopolymer compressed repeat copies, and one with uncompressed copies. Through the homopolymer compression, we hope to reduce the number of false negatives when we search the identifiers in the set of PacBio HiFi reads, since most errors in these reads relate to homopolymer run lengths. Figure 17 shows the lengths of the shortest identifiers from the copies in both datasets. As it turns out, about 100 copies from the repeat selection model output have identifiers that are shorter than 50 base pairs. The great majority of copies has identifiers that are shorter than 500 bp. This is especially true for the

compressed copies, where no identifier is longer than 1000 bp. We can thus say that, given the low error rate of PacBio HiFi reads, false negatives are unlikely, especially for the compressed identifiers. Regarding the 32 different T2T copies, the situation is a bit different. Again, the majority of the copies has identifiers that are no longer than 1000 bp. However, there are five outliers with identifiers that are significantly longer. In the most extreme case, the shortest identifier is about 22000 bp (uncompressed copies), or 16000 bp (compressed copies) long. Thus, for some T2T copies, the shortest identifiers are probably too long to trace them even in the highly accurate HiFi reads.

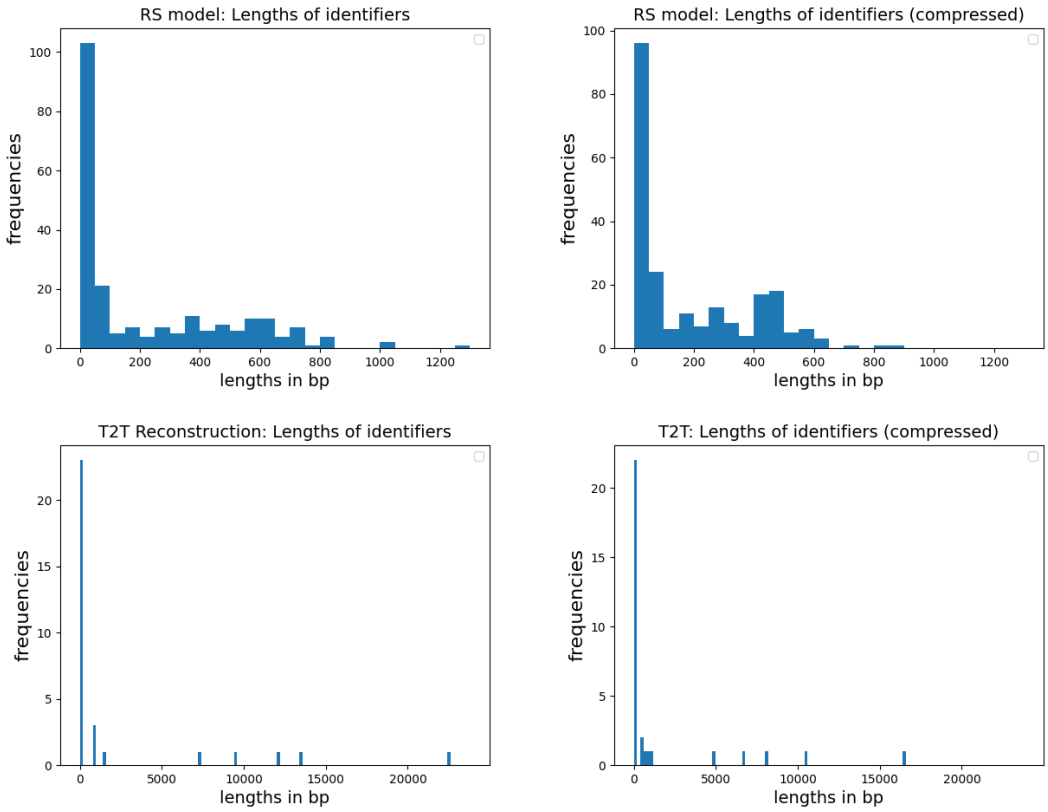

Figure 17: Histograms that show the lengths of the shortest identifiers for all repeat copies. The top row shows the identifier lengths in the copies from the repeat selection model. The bottom row shows the lengths for the 32 different T2T copies. The plots in the left column show the lengths for identifiers in the uncompressed copies. The plots in the right column show the lengths for the compressed copies.

For the repeat copies from our model, we searched all compressed and uncompressed identifiers in the sets of compressed and uncompressed PacBio HiFi reads, respectively. For 182 out of 221 uncompressed identifiers, and 183 compressed identifiers, we could find at least one match in the reads. Thus, for the majority of the copies generated by our model, there is at least some supporting evidence. As mentioned before, all repeat copies produced by the repeat selection model were unique, even though it is also possible for the model to select non-unique copies. For 143 uncompressed identifiers, and 160 compressed identifiers, we could find at least 10 matches in the reads. We thus see more identifiers with a higher number of matches when we include the homopolymer compression, what indicates that this additional

step improves the quality of our results. Interestingly, there are also 59 uncompressed, and 49 compressed identifiers that occur 100 times or more in the read data. This indicates that for some identifiers, there is more evidence in the data than our model can explain. These might occur more often in the real DNA, than the repeat selection model predicts. It is also interesting that fewer compressed than uncompressed copies occur more than 100 times in the data, indicating that runs with uncompressed data might also lead to overrepresentation of some identifiers. Figure 18 shows the results for both runs, with the number of matches for each identifier plotted against the identifiers' length. It is visible that the compressed run (bottom row) tends to produce a higher number of matches for long identifiers.

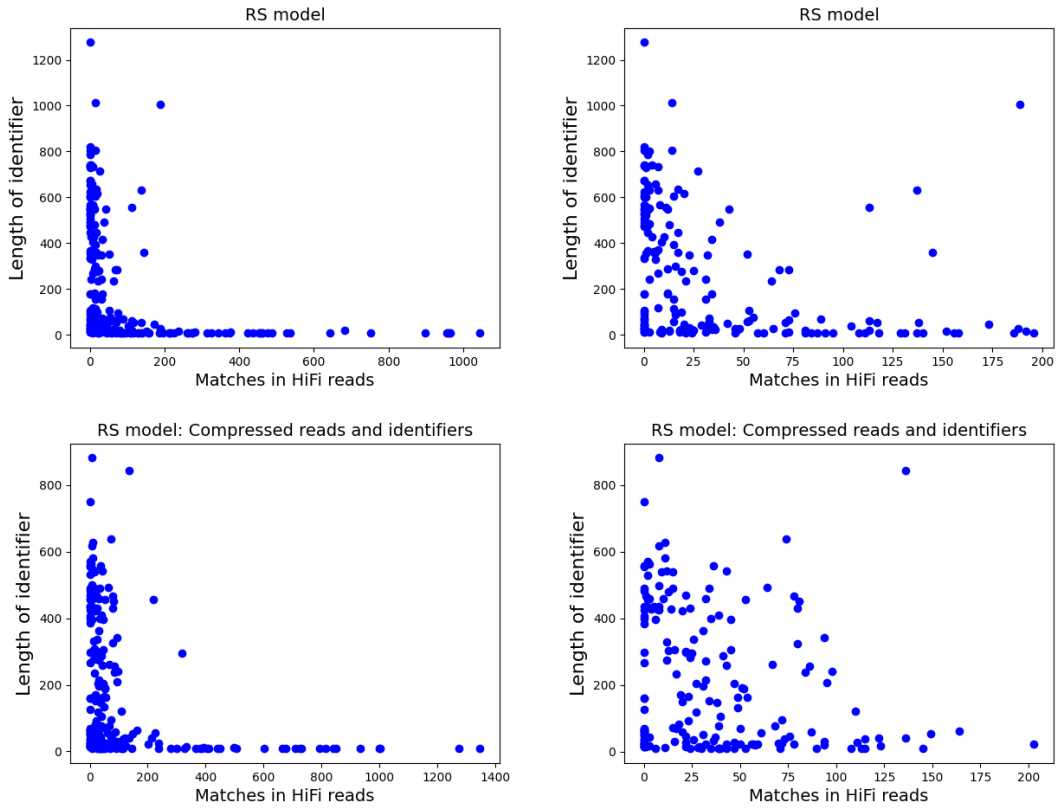

Figure 18: Length and number of matches in the HiFi reads for each identifier of the repeat copies that our model generated. The x-axis denotes the number of matches in the read dataset. The y-axis denotes the length of the identifiers. The top row shows the results for the uncompressed identifiers and reads, the bottom row shows the results for the compressed ones. The plots in the right column are zoomed-in versions of the plots in the left column, that show only identifiers with at most 200 matches.

For the 32 different T2T copies, we conducted the same experiments as for the copies from the repeat selection model. Figure 19 shows the results in the same way as figure 18 did for the previous run. For 27 out of 32 uncompressed identifiers, and 30 compressed ones, we could find at least one match in the HiFi reads. As the bottom right plot in figure 19 shows, homopolymer compression allowed us to find matches for three long identifiers with a length of 5000 bp or more. For 20 compressed and uncompressed identifiers, respectively, we could find more than 100 matches in the read data. This was expected, since many of the T2T copies

occur more than once on the repeat sites from the reference genome.

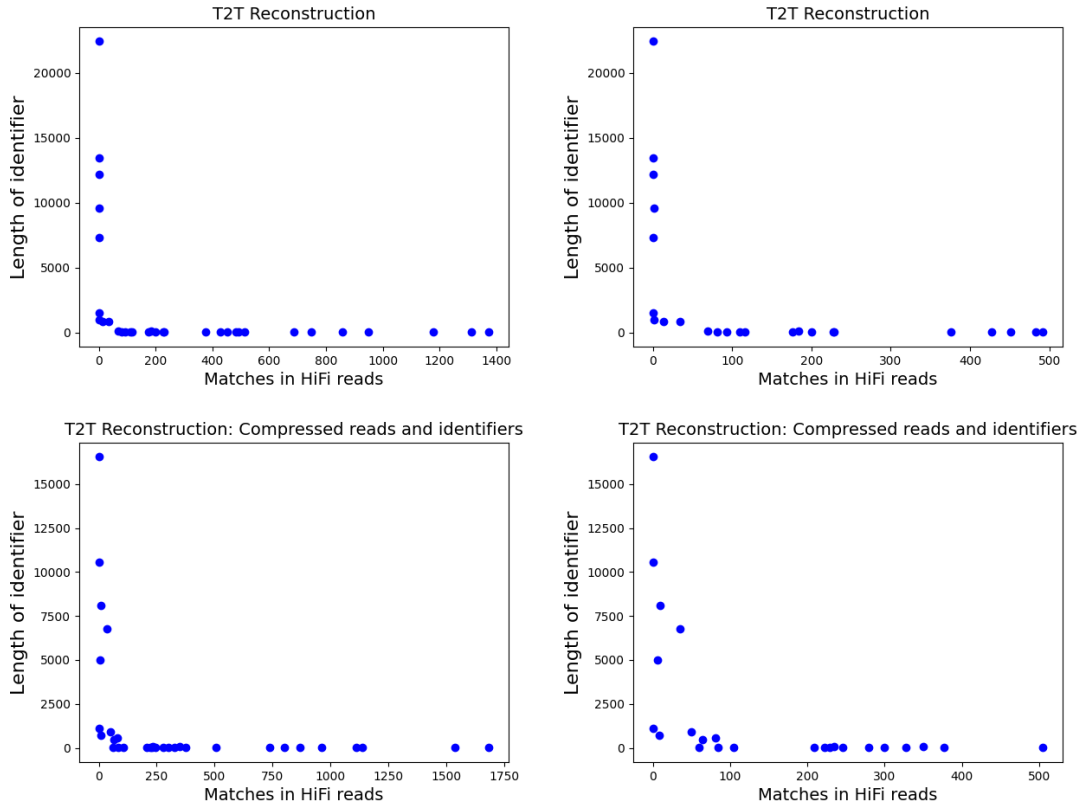

Figure 19: Length and number of matches in the HiFi reads for each identifier from the T2T copies.

On the whole, we assume that at least those 160 identifiers of the repeat copies from our model, that occur 10 times or more in the homopolymer compressed reads, are present in the real data. We assume it is very unlikely that the exact same, false sequence occurs multiple times in the highly accurate reads, and is also selected by our model as part of an optimal repeat copy. Thus, we conclude that a considerable part of the variation that our model predicts is genuine. At the same time, however, there are 38 identifiers for which there is no evidence. Also, 49 identifiers occur in the data much more often than we would expect based on the copies the repeat selection model produces. Thus, in total, it is still likely that the model generates too many variant copies. Regarding the T2T copies, many of the corresponding identifiers have a high number of matches in the read data. This indicates that it is justified to assume that these copies, or very similar ones, do indeed occur multiple times in the DNA. At the same time, as mentioned above, there is also evidence that supports many of the unique copies generated by the repeat selection model. We conclude that the extent of variation between individual repeat copies in the real genome must be higher than the T2T reference genome predicts, but lower than the repeat selection model predicts.

3.3 Reconstruction of rDNA repeats from HPRC samples

We used the repeat selection model to reconstruct the rDNA repeat copies from five human samples provided by the Human Pangenome Reference Consortium (HPRC). The samples we used are HG01258, HG01361, HG01952, HG02257, and HG03579. Read data for them is available online[5]. More information on the HPRC, and the quickly developing field of pangenomics in general, can be found in Miga et al. [35].

In all five cases, we started our workflow with the unmapped HiFi and ONT reads from the complete samples. We extracted the reads that align to the rDNA repeat sites by mapping all reads to the canonical rDNA unit KY962518.1. From the reads we found, we constructed blunted assembly graphs with MBG. Since these graphs contained two to three times as many nodes and edges as the assembly graph for CHM13, we decided to raise the k-mer and unitig abundance cutoffs from 10 to 24. The resulting, simpler graphs contain between 3142 and 5587 nodes, while the graph for CHM13 contains 4237 nodes. All five graphs showed a cyclic structure in their respective largest connected component. We broke the cycles in the same way we did for CHM13. On each graph, we had to remove about 10 edges. Aligning KY962518.1 to the graph for finding a suitable spot to break the cycle turned out slightly more difficult than in the case of CHM13. For HG01258, HG02257 and HG3579 it was not possible to find an end-to-end alignment for the reference unit with GraphAligner. This indicates that the rDNA repeats from the HPRC samples differ more strongly from the reference unit, than those from CHM13. However, we could align the *coding region* of KY962518.1 successfully in all cases. As the coding region marks the beginning of the rDNA reference unit, this is sufficient to find the right place for breaking the cycle. Incidentally, this experience also confirmed the assumption that the coding region is the most stable part of the rDNA repeats.

Regarding the sequence-to-graph alignments generated from the ONT reads, we decided to admit alignments into the set of potential repeats R_{Aln} if their length differs by at most 10% from the rDNA reference unit. This is more relaxed than in the case of CHM13, where we only allowed a length difference of 5%. This change is a reaction to the observations described in the last paragraph, which indicate that the repeats from the HPRC samples may be more different from the reference unit than those from CHM13. For determining k, the number of repeats to select, we proceeded in the same way as for CHM13. The same goes for the weight $w(v)$, which we again set to $log(|sequence(v)|)$ for each node v. For determining the coverage c_{avg} that a path explains on each node it traverses, we conducted runs with several different values for each sample. With binary search, we approximated an optimal value, that reduces the amount of overexplained and underexplained coverage as much as possible.

[5] *https://github.com/human-pangenomics/HPP_Year1_Data_Freeze_v1.0*

Table 1 lists up the parameter settings for all five samples, as well as the results of the runs with our model. Some columns in the table require a short explanation. **Coverage pre-run** denotes the total amount of weighted coverage on all nodes in the graph *before* we ran the model. This includes nodes that are not covered by any ONT alignment. **Coverage post-run** denotes the total amount of weighted, over- or underexplained coverage *after* we ran the model. For determining the percentage of **explained coverage**, we compute $\left(1 - \frac{\text{Coverage post-run}}{\text{Coverage pre-run}}\right) \cdot 100$. If the **gap** between the best solution and the best bound for a sample is smaller than 0.01%, we say that we have found an optimal solution.
For solving the Integer Linear Program, we used the commercial solver Gurobi 9.1.2 in all cases. Note that for all samples except HG01258, we could solve the OPTIMAL REPEAT SELECTION problem to optimality in less than 7500 seconds. In the case of HG02257, it took only 55 seconds to find an optimal solution. Only for HG01258, a small gap of 0.02% remained even after 7500 seconds. Since we could find optimal or near-optimal solutions quickly for the HPRC samples, we suspect that CHM13, where both COIN and Gurobi could not find provably optimal solutions in runs that took several days, is an exceptional case. We expect that, on most instances, an ILP based on our model can produce an optimal solution within a few hours. However, some uncertainty does remain, as runtime can vary by orders of magnitude between different instances.

Sample	$\lvert R_{Aln} \rvert$	**k**	c_{avg}	**Coverage pre-run**	**Coverage post-run**	**Explained coverage**	**ILP runtime**	**Gap**
HG01258	902	157	22	6843835.8	2523671.1	63.1%	7500s	0.02%
HG01361	689	112	27	5798865.7	2479513.9	57.2%	485s	$< 0.01\%$
HG01952	1485	152	27	7601852.0	2801403.8	63.1%	1826s	$< 0.01\%$
HG02257	397	124	21.5	4691861.1	1801916.7	61.6%	55s	$< 0.01\%$
HG03579	811	230	33	15272751.3	6014994.3	60.6%	544s	$< 0.01\%$

Table 1: Parameters and ILP results for the HPRC samples.

Regarding the coverage that is left over on the different instances after running the repeat selection model, further analysis is necessary to determine where it comes from. Based on the analysis of the CHM13 instance, we suspect that, again, some nodes are not covered by any ONT alignment and thus cannot be explained, while some others might be significanty overloaded by the model.

For the repeat copies from the HPRC samples, we created IGV-based visualizations in the same way as we did for the copies from CHM13. In all cases, we aligned the copies to the rDNA reference unit with minimap2, and ran IGV on the resulting alignment. Figures 20 and 21

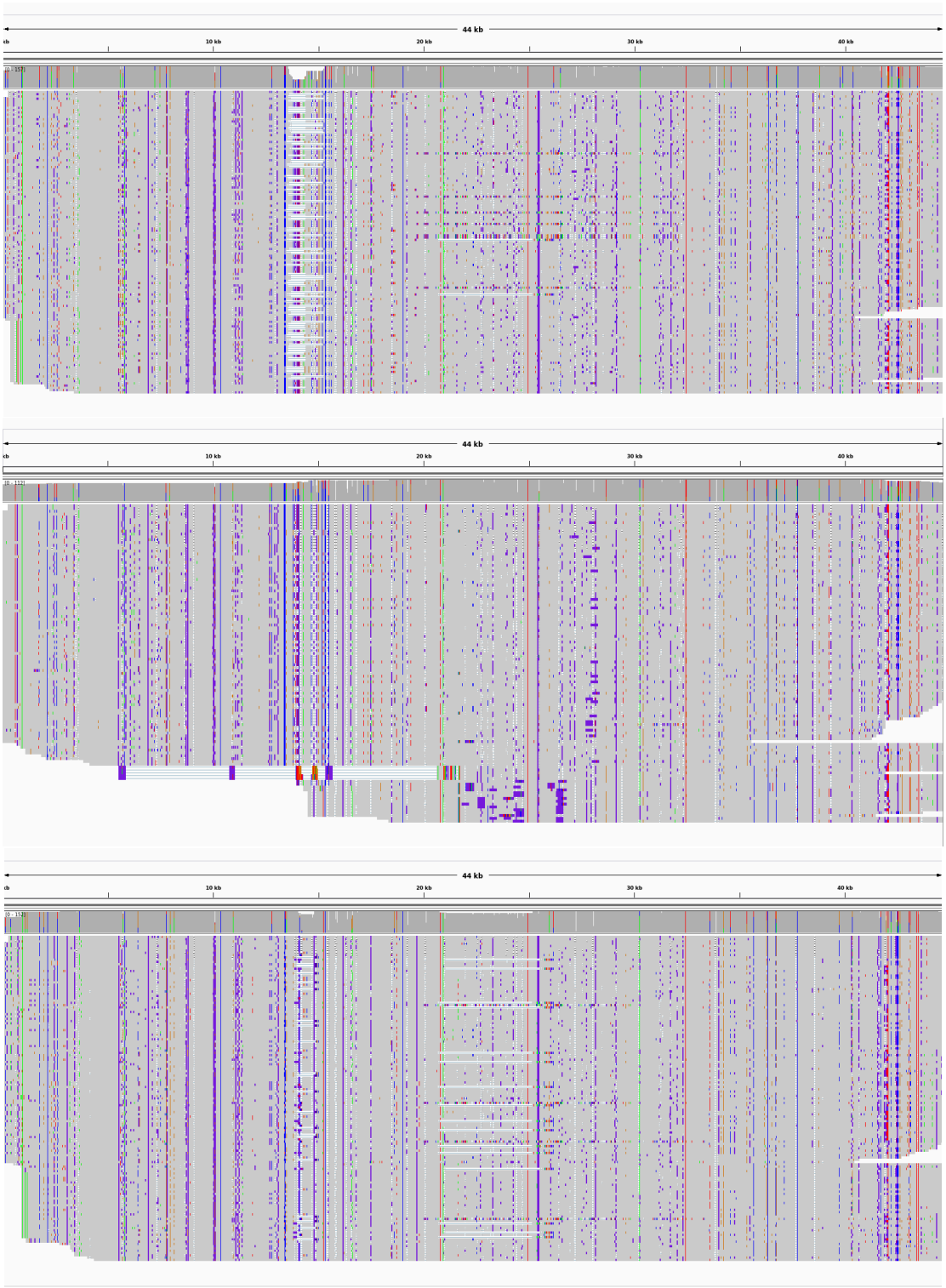

Figure 20: IGV visualizations for alignments of the repeat copies for HG01258 (top), HG01361 (middle) and HG01952 (bottom) against the rDNA reference unit.

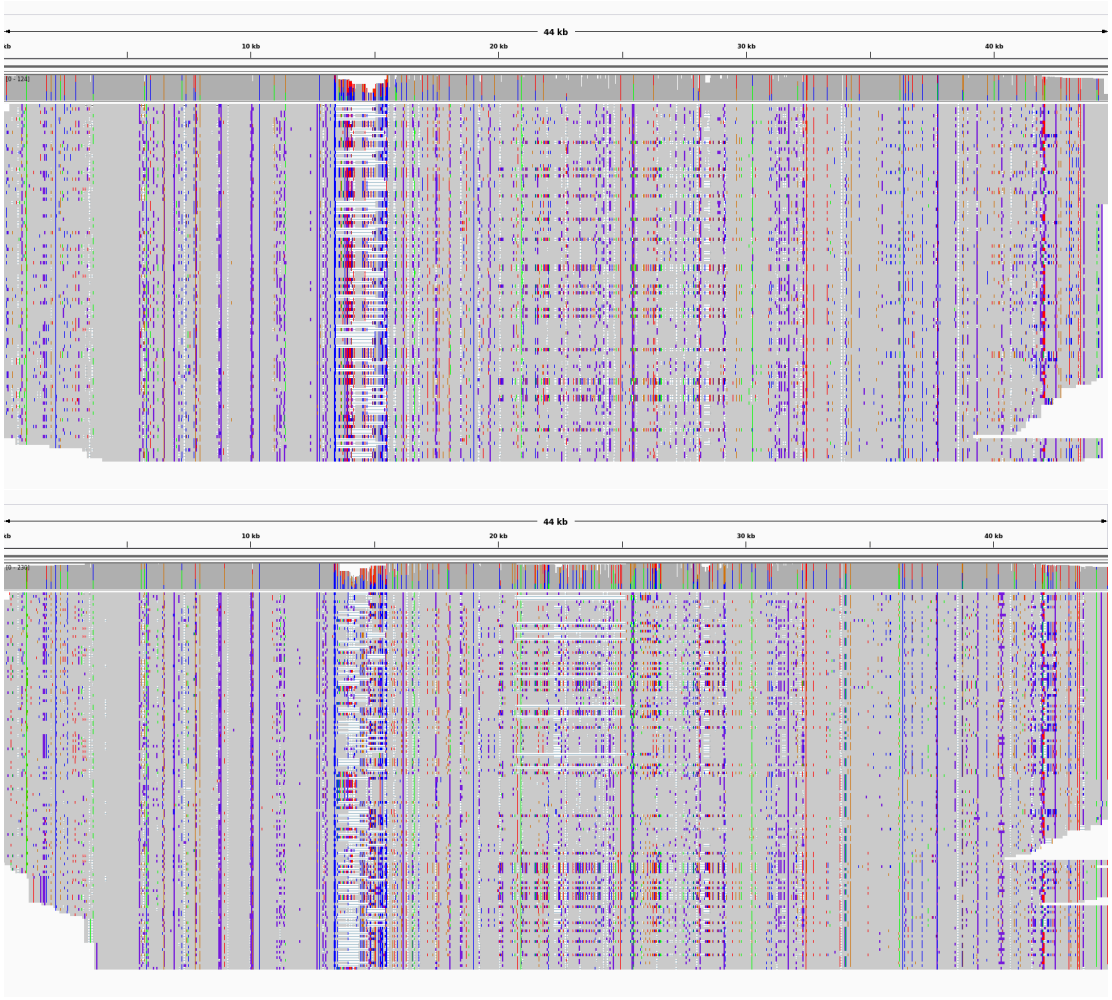

Figure 21: Visualized alignments for HG02257 (top) and HG03579 (bottom).

show these alignments. They reveal notable differences between the samples. For example, the deletions between coding and non-coding region are pronounced in HG01258, and even more so in HG02257 and HG03579, but mostly absent in HG01952. HG02257 and HG03579 also show a large number of polymorphisms in this region. For some copies from HG01361, minimap2 failed to align the first 15 to 20 kbp to the reference unit. This might indicate the presence of large-scale deletions, or other massive structural differences. The polymorphisms in the central part of the IGS, that we could observe in CHM13, are mostly absent from the HPRC samples. The only exception are some copies from HG01952, that show long deletions in this area.

4 Discussion

4.1 Evaluation of Results

We presented assemblies for the individual rDNA repeat copies of six different samples. This includes CHM13, as well as five samples provided by the Human Pangenome Reference Consortium. On all instances, we could either solve the OPTIMAL REPEAT SELECTION problem to optimality, or find near-optimal solutions with an approximation factor of 1.005 or better. In all cases, it was possible to find such a close approximation within 7500 seconds or less. The time for solving the problem to optimality on different instances does, however, vary greatly. The two most extreme cases were HG02257, where we found an optimal solution after 55 seconds, and CHM13, where we ran the solvers for several days, yet could not solve the problem to optimality. Despite this variance, we can confidently say that, for the instances we examined, it was always possible to find a close approximation of an optimal solution in an acceptable timespan. We expect this to be similar on other instances.

In order to assess the quality of our findings, we analysed the assembly results for the CHM13 instance, and compared them to the rDNA assembly conducted by the T2T Consortium. We could show that the results from both approaches differ considerably, especially in the sense that our model predicts greater variation between the individual copies. By retracing the shortest identifiers from each individual repeat copy in the HiFi read data, we could find evidence that a lot of the variance that our model predicts might indeed be present in the real DNA. However, we could also show that some identifiers of the copies from our model are not present in the HiFi data, while others are present significantly more often than the repeat selection model predicts. Thus, the model probably overestimates the variation between rDNA copies in the real genome by some degree.

For the HPRC assemblies, we did not analyse the results in the same detail as for CHM13. In part, this is due to time constraints. We focused our analysis on CHM13, because it serves as the basis for the T2T Consortium's reference genome. It thus offers a very convenient opportunity for comparing our output with state-of-the-art results. We expect a shortest identifier-based analysis of the HPRC samples to yield similar results compared to CHM13. Since a divergence in results could tell us something about what weak points our model may have, and on which instances it works particularly good or bad, we consider such an analysis to be a good starting point for future attempts to overhaul and improve the model.

What we did do with the HPRC samples was creating visualizations based on alignments against the canonical rDNA unit. These visualizations serve to uncover structural differences between the different copies from each individual sample, as well as between the complete samples. Through the visualizations, we could show that our model predicts large deletions and insertions for a subset of copies from all of the samples. Most of these features are located

on the same hotspots, for example the junction between the coding and non-coding region of the copies. From sample to sample, the extent and frequency of these large-scale deviations from the canonical rDNA unit differ strongly. The same is true for single-nucleotide polymorphisms (SNPs). For these mutations, the model predicts some recurring hotspots. One is located at the end of the copies, and another one in the region between 20 and 32 kbp. Some other ones are present in the coding region. While the hotspots in the coding region and at the end of the copies are present in all samples, the region between 20 and 32 kbp is less active in some of the samples, especially HG01258, HG01361 and HG01952. The presence of large-scale structural differences between individual rDNA copies, and between different samples, is, in principle, consistent with what is known from the rDNA repeat copies (see especially Smirnov et al. [5]). Our model might even underestimate the copies' variance in length, since we restricted the set of potential paths on the graph, that we select from, to paths whose corresponding sequences are similar in length to the canonical rDNA unit. However, there is also some disagreement with the literature regarding where exactly the most variant parts of the copies are located. See figure 1 in Smirnov et al. [5] for a visual comparison. One suggestion is that this difference might result from analysing samples that come from different populations. We have not included population data into our present analysis, but a comparison between the alignments of the HPRC samples in section 3 shows that the extent and location of the variability hotspots can vary to some degree between different samples. A future study with a larger sample size, that takes ancestral data for different individuals into account, might serve to examine this assumption further.

On the whole, we believe that our model is able to produce viable results for a diverse range of samples, and in an acceptable timeframe. Yet, there is also room for improvement. One important point is that, in the preprocessing, we currently have to remove edges manually to break the cyclic structure of the assembly graph's largest connected component. It would be a clear improvement if we were able to automate this step. Also, the current depencence of our model on a *blunted* assembly graph may be disadvantageous in some cases, since it is hard to assess in what way redundant nodes, that are created in the bluntification process, may distort the results. Another point is the decision-making regarding the parameter settings. Right now, the methods for determining k and $w(v)$ have, to some degree, an ad hoc character. In the case of c_{avg}, we tune the parameter by running the model a number of times, to search a value that leads to the least amount of unexplained coverage. Stricter criteria for choosing these values would be welcome.

More points of potential improvement can be derived from the analysis of the model's output. This relates especially to the findings from sections 3.1.2, 3.1.3 and 3.2.3. We know that the model overloads some of the nodes in the assembly graph significantly, while some other nodes are not covered by any of the ONT alignments that serve as potential paths. Even though this mostly affects shorter nodes with low coverage, we need to get a more precise understanding of how this comes about, in order to assess possibilities for improvement. Moreover, from the

shortest identifier-based analysis of the output, we know that the model tends to overestimate the variation between the different rDNA copies in the genome. Again, we need to know more about why this happens, in order to make suitable adjustments. One possible improvement might consist in setting higher k-mer and unitig abundance cutoffs for the assembly graph, what would lead to fewer nodes and edges, and hence fewer possibilities for variant paths.

4.2 Future Work

As outlined in the last paragraph, there are a number of possibilities for improving the work we presented in this thesis, and a revision of the current model is a worthy future goal. Aside from this, there is another, bigger challenge. In section 1.3, we pointed out that the model we present in this thesis only yields an assembly of *individual* rDNA repeat copies. This allows for comparing the structure and variance of rDNA repeats within a sample, and between samples. Moreover, for now, the number of rDNA copies we assemble is based on the assumption that we reconstruct a haploid genome. Real human genomes, as is well known, are diploid, while other organisms can have even higher ploidy. For a complete reconstruction of the rDNA repeat sites, two more steps are therefore necessary. The first one consists in partitioning the repeat copies according to their chromosomes of origin, and resolving the order of the copies on the chromosomes. The second one consists in haplotyping the copies. This means, we need to move from a reference genome-like model of the repeat sites, that yields one idealized haplotype for each of the five sites, to an assembly that includes two different haplotypes for each chromosome. Note that it is possible that not only the individual copies, but also the number of copies on a given chromosome, might differ between the two haplotypes.

We now want to present a first sketch of a method for partitioning the repeat copies onto the five different chromosomes, as well as for resolving the order of the copies on the chromosomes. It is based on solving a variant of the TRAVELING SALESMAN problem. As input, the proposed method takes the set of assembled copies from the repeat selection model, and the unaligned ONT reads from the repeat sites. The main idea consists of the following steps:

1. For each pair of repeat copies, i and j, cut both copies in half, and create two sequences s_{ij} and s_{ji} out of the halves. The first half of s_{ij} is the second half of copy i, and the second half of s_{ij} is the first half of copy j. Conversely, the first half of s_{ji} is the second half of copy j, and the second half of s_{ji} is the first half of copy i.

2. Map the ONT reads against each s_{ij}, to find out how much support there is for the sequences, and thus, how plausible it is that the corresponding copies are neighbors in the genome. To assess the plausibility for each sequence s_{ij}, we need to develop a function that assigns a cost c_{ij} to each s_{ij}, based on its support in the ONT reads. It may be difficult to determine what 'good support' in the error-prone ONT reads means, and to find a suitable cost function.

3. As we have seen, the repeat selection model yields a set of k repeat copies. Step 1 of the method outlined here yields a set of $2k^2$ sequences. Step 2 yields a cost c_{ij} for each sequence s_{ij}. From this input, construct a complete, bidirectional graph, where the nodes correspond to the repeat copies, while the edges correspond to the sequences. Thus, for repeat copies i and j, the edge (i, j) corresponds to s_{ij}, while (j, i) corresponds to s_{ji}. Each edge (i, j) is labeled with the cost c_{ij} of the corresponding sequence.

4. On the graph from step 3, find a minimum-cost tour that traverses all nodes exactly once. We allow 4 wildcards, that is, four edges whose cost can be set to 0. These edges can be chosen freely from the set of all edges in the graph. The plan is that the wildcards separate the arrays of copies on the different chromosomes. Each subsequence of nodes from the optimal tour, that contains no wildcard, corresponds to a repeat array on one chromosome. We think this makes sense because, given that the copies are located in arrays on five different chromosomes, the unknown tour that corresponds to the ground truth should contain subsequences of very low cost, separated by four expensive edges. The wildcards should allow for eliminating the cost of these expensive edges, such that we can find an optimal tour that corresponds to what is present in the DNA.

5. Given the five arrays of repeat copies that step 4 yields as output, assign each of them to one of the five chromosomes in question. Currently, we do not have a suitable criterion for the assignment, yet we believe that it is possible to construe one.

It is clear that this is only a very rough sketch of a method that might lead to success, and there are a number of problems. Especially, it might be difficult to construe a meaningful cost function for the sequences that serve as edges in the graph. However, we think that a good solution to the problem of partitioning the copies into five sets, and resolving their order, is possible with a procedure that goes along these lines.

After assigning the copies to the chromosomes, and ordering them, we still need to make the transition from a haploid to a diploid assembly of the rDNA repeat sites. This means, we need to determine which rDNA copies from a given chromosome are located on the same copy of that chromosome. As of yet, we have not looked into possible solutions to this problem, or its connections to the problem of partitioning and ordering the repeat copies. On the way towards an automated method for complete *de novo* assembly of rDNA repeat sites, it presumably marks the final step.

5 Conclusion

In this thesis, we presented a model for the assembly of rDNA repeat copies on human genomes. We proved the hardness of the optimization problem that we solve, and used our model to assemble the rDNA repeat copies of six different human samples. On the whole, we believe that our approach is successful, while improvements are certainly possible. Importantly, a comparison between the output of our model, and the T2T Consortium's assembly of CHM13, showed that our model generates repeat copies that vary more strongly than the ones from the T2T assembly. Some, though not all, of the additional variation appears to be genuine. We need to analyse the output of our model further, in order to better understand where its specific properties come from. Additionally, we need to collect more evidence for or against the justifiability of the model's output on different samples, in order to optimize its results.

With regard to runtime, we could show that our model is able to provide good approximations, and often even optimal solutions, in a timespan of less than two hours. Time investment for the preprocessing steps is harder to calculate. Based on our experience, it varies only slightly between different samples, though. On the whole, we believe that our method is fast enough to allow for the reconstruction of, e.g., a few dozens of samples in an acceptable timeframe. We hope that we can, for example, apply it to study the differences in the rDNA repeats of different human populations.

Future challenges include partitioning the repeat copies onto their five chromosomes of origin, and resolving their order on the chromosomes, as well as reconstructing the two different haplotypes of human samples. We hope that, for these steps, it is also possible to develop feasible, combinatorial optimization-based methods.

6 Literature

References

[1] Saumya Agrawal and Austen R.D. Ganley. “The conservation landscape of the human ribosomal RNA gene repeats.” In: *PLoS ONE* 13 (12 2018). DOI: 10.1371/journal.pone.0207531.

[2] Meng Wang and Bernardo Lemos. “Ribosomal DNA harbors an evolutionarily conserved clock of biological aging.” In: *Genome Research* 29 (3 2019), pp. 325–333. DOI: 10.1101/gr.241745.118.

[3] Elena M. Malinovskaya et al. “Copy Number of Human Ribosomal Genes With Aging: Unchanged Mean, but Narrowed Range and Decreased Variance in Elderly Group.” In: *Frontiers in Genetics* 9 (306 2018). DOI: 10.3389/fgene.2018.00306.

[4] Matthew M. Parks et al. “Variant ribosomal RNA alleles are conserved and exhibit tissue-specific expression.” In: *Science Advances* 4 (2 2018). DOI: 10.1126/sciadv.aao0665.

[5] Evgeny Smirnov et al. “Variability of Human rDNA.” In: *Cells* 10 (2 2021). DOI: https://doi.org/10.3390/cells10020196.

[6] Sergey Nurk et al. “The complete sequence of a human genome.” In: *Science* 376 (6588 2022), pp. 44–53. DOI: 10.1126/science.abj6987.

[7] Aaron M. Wenger et al. “Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome.” In: *Nature Biotechnology* 37 (2019), pp. 1155 –1162. DOI: 10.1038/s41587-019-0217-9.

[8] Miten Jain et al. “Nanopore sequencing and assembly of a human genome with ultra-long reads.” In: *Nature Biotechnology* 29 (4 2018), pp. 338 –345. DOI: 10.1038/nbt.4060.

[9] Mikko Rautiainen and Tobias Marschall. “MBG: Minimizer-based sparse de Bruijn Graph construction.” In: *Bioinformatics* 37 (16 2021), pp. 2476–2478. DOI: 10.1093/bioinformatics/btab004.

[10] Mikko Rautiainen and Tobias Marschall. “GraphAligner: rapid and versatile sequence-to-graph alignment.” In: *Genome Biology* 21 (253 2020). DOI: 10.1186/s13059-020-02157-2.

[11] Mitchell R. Vollger et al. “Segmental duplications and their variation in a complete human genome.” In: *Science* 376 (6588 2022). DOI: 10.1126/science.abj6965.

[12] Jasmijn A. Baaijens et al. “Full-length de novo viral quasispecies assembly through variation graph construction.” In: *Bioinformatics* 35 (24 2019), pp. 5086 –5094. DOI: 10.1093/bioinformatics/btz443.

[13] Jasmijn A. Baaijens, Leen Stougie, and Alexander Schönhuth. “Strain-Aware Assembly of Genomes from MixedSamples Using Flow Variation Graphs.” In: *RECOMB 2020 - 24th International Conferenceon Research in Computational Molecular Biology* (2020), pp. 221 –222. DOI: 10.1007/978-3-030-45257-5_14.

[14] Romeo Rizzi, Alexandru I. Tomescu, and Veli Mäkinen. “On the complexity of Minimum Path Cover with Subpath Constraints for multi-assembly.” In: *BMC Bioinformatics* 15 (S5 2014). DOI: 10.1186/1471-2105-15-S9-S5.

[15] Ryan R. Wick et al. “Bandage: interactive visualization of de novo genome assemblies.” In: *Bioinformatics* 31 (20 2015). DOI: 10.1093/bioinformatics/btv383.

[16] Heng Li. “Minimap2: pairwise alignment for nucleotide sequences.” In: *Bioinformatics* 34 (18 2018), pp. 3094 –3100. DOI: 10.1093/bioinformatics/bty191.

[17] James T. Robinson et al. “Integrative Genomics Viewer.” In: *Nature Biotechnology* 29 (1 2011), pp. 24 –26. DOI: 10.1038/nbt.1754.

[18] Heng Li et al. “The Sequence Alignment/Map format and SAMtools.” In: *Bioinformatics* 25 (16 2009), pp. 2078 –2079. DOI: 10.1093/bioinformatics/btp352.

[19] Heng Li. “Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences.” In: *Genome Biology* 32 (14 2016), pp. 2103 –2110. DOI: 10.1093/bioinformatics/btw152.

[20] Jordan M. Eizenga et al. “Walk-Preserving Transformation of Overlapped Sequence Graphs into Blunt Sequence Graphs with GetBlunted.” In: *Conecting with Computability. CiE 2021*. (2021), pp. 169–177. DOI: 10.1007/978-3-030-80049-9_15.

[21] Heng Li, Xiaowen Feng, and Chong Chu. “The design and construction of reference pangenome graphs with minigraph.” In: *Genome Biology* 21 (265 2020). DOI: 10.1186/s13059-020-02168-z.

[22] S. B. Needleman and C. D. Wunsch. “A general method applicable to the search for similarities in the amino acid sequence of two proteins.” In: *Journal of Molecular Biology* 48 (3 1970), pp. 443 –453. DOI: 10.1016/0022-2836(70)90057-4.

[23] Kun-Mao Chao, William R. Pearson, and Webb Miller. “Aligning two sequences within a specified diagonal band.” In: *Bioinformatics* 8 (5 1992), pp. 481 –487. DOI: 10.1093/bioinformatics/8.5.481.

[24] Esko Ukkonen. “Algorithms for approximate string matching.” In: *Information and Control* 64.1 – 3 (1985), pp. 100 –118. DOI: 10.1016/S0019-9958(85)80046-2.

[25] Quiang-Sheng Hua et al. “Exact Algorithms for Set Multicover and Multiset Multicover Problems.” In: *International Symposium on Algorithms and Computation* (2009), pp. 34–44. DOI: 10.1007/978-3-642-10631-6_6.

[26] Richard M. Karp. “Reducibility among Combinatorial Problems.” In: *Complexity of Computer Computations. The IBM Research Symposia Series.* (1972), pp. 85–103. DOI: https://doi.org/10.1007/978-1-4684-2001-2_9.

[27] Veli Mäkinen et al. *Genome-Scale Algorithm Design. Biological Sequence Analysis in the Era of High-Throughput Sequencing*. 1st. Cambridge University Press, 2015. ISBN: 978-1-107-07853-6.

[28] Vladimir I. Levenshtein. “Binary Codes Capable of Correcting Deletions, Insertions and Reversals.” In: *Soviet Physics Doklady* 10 (8 1966), pp. 707 –710.

[29] Udi Manber and Gene Myers. “Suffix Arrays: A New Method for On-Line String Searches.” In: *SIAM Journal on Computing* 22 (5 1993), pp. 935 –948. DOI: 10.1137/0222058.

[30] Ge Nong, Sen Zhang, and Wai Hong Chan. “Linear Suffix Array Construction by Almost Pure Induced-Sorting.” In: *2009 Data Compression Conference* (2009). DOI: 10.1109/DCC.2009.42.

[31] Fei Shi. “Suffix arrays for multiple strings: A method for on-line multiple string searches.” In: *Concurrency and Parallelism, Programming, Networking, and Security. ASIAN 1996* (1996), pp. 11 –22. DOI: 10.1007/BFb0027775.

[32] Toru Kasai et al. “Linear-Time Longest-Common-Prefix Computation in Suffix Arrays and Its Applications.” In: *Combinatorial Pattern Matching. CPM 2001.* (2001), pp. 181 –192. DOI: /10.1007/3-540-48194-X_17.

[33] Paniz Abedin, M. Oguzhan Külekci, and Shama V. Thankachan. “A Survey on Shortest Unique Substring Queries.” In: *Algorithms* 13 (9 2020). DOI: 10.3390/a13090224.

[34] John Ellson et al. “Graphviz and dynagraph - static and dynamic graph drawing tools.” In: *Graph Drawing Software*. Springer-Verlag, 2003, pp. 127–148. DOI: 10.1007/978-3-642-18638-7_6.

[35] Karen H. Miga and Ting Wang. “The Need for a Human Pangenome Reference Sequence.” In: *Annual Review of Genomics and Human Genetics* 22 (2021), pp. 81–102. DOI: 10.1146/annurev-genom-120120-081921.