De novo approaches to haplotype-aware genome assembly

Jasmijn Anne Baaijens

Title: De novo approaches to haplotype-aware genome assembly
Author: Jasmijn A. Baaijens
Cover illustration: TheDigitalArtist at www.pixabay.com
Production: Gildeprint
ISBN: 978-94-632-3743-7

The research described in this thesis was conducted at Centrum Wiskunde & Informatica (CWI) and financially supported by the Netherlands Organization for Scientific Research (NWO) through Vidi grant 679.072.309.

De novo approaches to haplotype-aware genome assembly

Algoritmen ten behoeve van haplotype assemblage zonder referentiegenoom

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof.dr. H.R.B.M. Kummeling, in gevolge het besluit van het college voor promoties in het openbaar te verdedigen op woensdag 25 september 2019 des ochtends om 10.30 uur

door

Jasmijn Anne Baaijens

geboren op 10 december 1990 te Alphen aan den Rijn

Promotor: Prof. dr. A. Schönhuth

Table of contents

Preface

When I started my PhD research in September 2014, I was surprised by the broad variety of research topics in bioinformatics and the speed by which this field develops. After spending two months on reading a bit of everything, my attention was drawn towards the haplotype-aware genome assembly problem. I found it remarkable that, at that time, there were so few methods available for haplotype reconstruction *without a reference genome*. In particular, the problem of *de novo viral quasispecies assembly* had not yet been resolved.

Intrigued by these computational challenges and the 2014–2016 outbreak of Ebola virus disease in West Africa, the primary goal of my research became to develop a de novo approach to viral quasispecies assembly. At first, it seemed that for every problem that we solved, ten new issues arose. But after two years of research, we were ready to present our first results. And now, after four years of hard work, I am proud to write that we can present a full, *de novo* solution to the viral quasispecies problem.

This thesis bundles the research papers presenting the results we achieved, the scope of which is wider than viral quasispecies assembly alone. Although the core chapters are rather technical, the first chapter gives a mild introduction to the field of haplotype-aware genome assembly. Besides framing the problem and providing some context, I hope that this chapter will give many people that I hold dear a further understanding of what I have been working on all this time.

Acknowledgements

These four years of PhD research have been an amazing experience, which would not have been possible without the support of many people.

First and foremost, I would like to thank my supervisor and promotor Alexander Schönhuth, for giving me the chance to discover the world of bioinformatics. I am very grateful for all the opportunities you have given me, opening up your network and giving me the responsibility and confidence to present our work at several conferences. The visit to Montpellier just three months after the beginning of my PhD was the perfect kick-off for the viral quasispecies project. You were always supportive and understanding if

something did not work as planned, yet challenged me when I needed it. So often have I heard “You may not like it, but ...”, knowing that you were right. The freedom you gave me to pursue my own interests has allowed me to grow as a researcher. I enjoyed my independence, yet whenever I needed guidance you were there to assist. Thank you for everything, I am looking forward to meet again in any future endeavors.

Although it has been a long way, these years of work certainly have not been lonely. I have had the pleasure of working with many great people. I would like to thank Eric Rivals and Amal Zine El Aabidine for the fruitful collaboration on the SAVAGE project. Also, I would like to thank Christopher Esterhuyse for the work on suffix-prefix overlaps. It was a delight to guide you through your bachelor’s project and your work has made a big contribution to this thesis. To Bastiaan van der Roest, Johannes Köster, and Leen Stougie, I am thankful for the great teamwork on Virus-VG. Bastiaan, you made such a great start on this project. Johannes, thank you for always being ready to help, even after you left CWI. Leen, I enjoyed working together with you a lot: your enthusiasm is contagious and your help has been invaluable.

I would like to thank all members of the Life Sciences (& Health) group in the past four years that have made my time at CWI such an enjoyable one. In particular Lisanne, thank you for all the nice chats, walks, and pingpong breaks. Sharing an office with you was really great. In addition, a special thanks to Davide and Mathé, for many pleasant coffee breaks. Also to Marleen and Vincent, working with you has been a lot of fun and a welcome distraction from the thesis work. And, as promised, my thanks to Peter, not only for sharing your computing power in times of need, but also for your general support.

CWI has a great working environment, powered by a fantastic support staff. Michael and Maarten, I really appreciate all your help with IT-related issues; if I would have tracked all my movements through CWI, the walk from my office to yours would stand out. I would also like to express my gratitude to Nada, for your kind help with simply everything. A great addition to the fun I had at CWI have been the PhD activities, at the receiving end as well as on the organizing end. A special thanks to the past and current members of the activity committee. In particular to Léon, thank you for making these events possible through your support and involvement.

Finally, I am truly grateful for all the support I have had from friends and family. I am greatly indebted to my parents, without whom this thesis would never have been finished (or even started). And, most importantly, my dear husband Michiel, you have contributed so much to this thesis by your continuous love and support, and by providing a peaceful and happy home.

Jasmijn Baaijens
Alphen aan den Rijn, May 2019

CHAPTER 1

INTRODUCTION

1.1 DNA, RNA, and genetic variation

DNA and RNA molecules are found in abundance in all known forms of life. One or more of these molecules together form its owner's *genome*, storing all information needed to build and maintain the organism: a genetic blueprint. The overall term for DNA and RNA is *nucleic acids* and they consist of chains of *nucleotides*, the building blocks of all genetic information. A nucleotide consists of three components: a sugar, a phosphate group, and a nitrogenous base. There are five primary nitrogenous bases—adenine (A), cytosine (C), guanine (G), thymine (T), and uracil (U)—giving rise to five different nucleotides, also referred to as *bases*. The order of nucleotides in DNA and RNA ultimately contains the information for the hereditary and biochemical properties of life.

The nucleotides in DNA are represented by letters from the four letter alphabet $\{A, C, T, G\}$. Each of these bases can form hydrogen bonds with its opposing base: A with T and C with G. These pairs of bases are called *complementary*. A DNA molecule typically consists of two chains of nucleotides, called *strands*, which coil around each other to form a *double helix*. The two strands run in opposite directions, they are said to be *reverse complements*. Both strands store exactly the same information; this redundancy enables repair mechanisms to correct errors in case of DNA damage. In RNA, the thymine base (T) is substituted by uracil (U) and the complementary base pairs become A with U and C with G. However, RNA is usually single stranded, which makes it more vulnerable to corruption than DNA. Many viral genomes are encoded as RNA, but this type of nucleid acid can also serve as a communication step that directs protein synthesis. In this process called *translation*, the RNA sequence determines the amino acid sequence of the protein that is being produced.

Many genomes come in copies, where each copy stems from one of the ancestors. The number of copies determines the *ploidy* of the organism: *haploid* denotes a single copy, while *diploid* relates to two copies, and *polyploid* refers to more than two copies (depending on the context, polyploid may also include diploid). For example, the human genome is encoded as DNA within 46 chromosomes which come in 23 pairs, one copy from each parent. A haploid human genome (found in germ cells) counts three billion base pairs, while a diploid human genome has twice as many base pairs.

Genomes within a population show *genetic variation* as a result of mutation and recombination; the variants of a given gene are called *alleles*. Also the copies of the genome within a single individual will differ in terms of the alleles present. Genetic differences between individuals play an important role in evolution, as genomic alterations can affect gene expression levels and enable the development of novel gene functions. *Mutations* can be divided into three classes: single nucleotide polymorphisms (SNPs), insertions or deletions, and structural variations (SVs) such as duplications or inver-

sions. Another source of genetic variation is *recombination*, where the different copies of the genome within a single cell are recombined. Mutation and recombination rates can vary from species to species and even from position to position within a single genome [87].

1.2 Genome sequencing

The order of nucleotides within a genome can be determined through *sequencing*. However, it is hard to read the whole genome all at once; available methods can only read stretches of nucleotides of limited length. The fragments produced by a sequencing machine are referred to as *reads*. In general, the reads come together with *quality scores* indicating per-base error probabilities. By producing multiple copies of the genome through *amplification* and randomly breaking each copy into readable fragments, overlaps between reads are created that enable reconstruction of the original genome. The number of copies determines the average amount of reads covering a given position of the genome; this value is referred to as *sequencing depth* or *coverage*. Different technologies produce reads of different lengths and error profiles, at varying costs and production times.

The first generation of sequencing methods were manual processes, of which the most common technique is known as Sanger sequencing [106]. This technique has been available since 1977 and has been used widely since. The Sanger method produces sequencing reads with lengths up to 1000 bases and average error rates of only 0.001% [124], but the sequencing process is rather expensive (in terms of both time and money) when it comes to sequencing entire mammalian genomes. Despite its high costs per base, Sanger sequencing is still used to date for small-scale experiments, where it operates efficiently and achieves low error rates.

Then, around the year 2005, a new era of genome sequencing started with the advent of *next-generation sequencing* (NGS). New technologies became available that were able to speed up the sequencing process tremendously, while maintaining much lower production costs as well. The downside, however, being that these machines were less reliable than Sanger sequencing, with error rates of 0.1–1.0% at least 100-fold higher than before [109, 124]. Moreover, with read lengths varying between 36 and 600 bases, reads are significantly shorter than those obtained with Sanger sequencing. In order to deal with increased error rates, genomes are sequenced at increased coverage. This creates redundancy in the data and thereby allows for correction of sequencing errors (up to a certain degree).

Since 2008, yet another range of sequencing technologies—described as *third-generation sequencing* (TGS)—has emerged. These technologies are characterized by much longer read lengths and are therefore also referred to as *long read sequencers*.

Depending on the technology used, read lengths can reach up to 900 kilobases [53]. But again, this advance comes at the cost of significantly increased error rates: initially these were as high as 20%, but over the years have been reduced to 13–15% in 2017 and are expected to be reduced even further [109, 126].

Although long read sequencing is very promising, sequencing read archives have been filled with huge amounts of NGS data. The algorithms presented in this thesis were designed for NGS data sets, in particular data obtained with Illumina sequencing machines. However, we have kept a clear view towards the future: minor modifications should enable processing of TGS reads (see also Chapter 6).

1.3 Genome assembly

The process of reconstructing a genome from sequencing data is called *genome assembly*. This is typically performed in two steps: first, the sequencing reads are used to build contiguous sequences of maximal length, referred to as *contigs*. Second, these contigs are linked together into sequencing of contigs, ordering (where possible) the contigs as they appear in the genome. The resulting creating chains of contigs are referred to as *scaffolds*.

Since the first sequencing techniques, many genomes have been assembled and databases have been filled with these sequences. Reference sequences have been built for various species, such as the human reference genome, which can serve as a guideline in new analyses. We distinguish between two classes of approaches to genome assembly: *reference-guided* and *de novo* (i.e. reference-free) assembly.

1.3.1 Reference-guided assembly

In reference-guided assembly, one or more known genome sequences are used to assemble the genome under consideration. Instead of reconstructing a sequence from scratch, we can align new sequencing reads to the existing genome sequence(s) and observe the differences; these processes are known as *read mapping* and *variant calling*, respectively. Although reference-guided assembly is computationally much more efficient than reference-free assembly, the disadvantage is that such an approach is only feasible if the existing assemblies are sufficiently similar to the genome to be assembled. The resulting assemblies often show a bias towards the reference genome(s) used. This bias can be reduced by using a collection of reference genomes that captures variation within a population, instead of single (linear) reference genome [31, 90].

1.3.2 De novo assembly

The alternative to reference-guided assembly is de novo assembly, where the genome is reconstructed from the sequencing data without the use of a reference genome. Sequence graphs are commonly used data structures in de novo assembly, in particular *de Bruijn graphs* [25, 94] and *overlap graphs* [83, 85]; in the mean time, these techniques have been perceived as assembly paradigms [84].

A de Bruijn graph stores the information from the reads in the form of k-mers, which are substrings of the reads of length k. Every node represents a k-mer that is present in the reads, and directed edges are drawn between any pair of nodes for which the k-mers have an exact suffix-prefix overlap of length $k-1$. In such a graph, assuming that there are no sequencing errors, the genome can be represented by a path visiting every edge exactly once [94].

In an overlap graph, every read is represented as a node, and edges indicate suffix-prefix overlaps between the sequences of the corresponding nodes. Edges are drawn for sequence overlaps of any length (above a certain threshold) and inexact suffix-prefix overlaps are allowed (up to a certain threshold). A specific type of overlap graph is the string graph, where any contained reads ("inclusions") and all transitive edges have been removed. In a world without sequencing errors, the genome is represented by the shortest possible path that visits all nodes of the string graph at least once [83].

Both of these paradigms have their advantages and disadvantages: while the problem formulation using de Bruijn graphs allows for more efficient solutions, some information is lost when decomposing reads into k-mers. Depending on the application, one paradigm may be preferable over the other; we will consider this choice in Chapters 2 and 3.

1.3.3 Error correction

Since sequencing reads contain errors, with error rates varying per sequencing technology, an important component of genome assembly is error correction. Some assemblers assume the input to be error-free by applying specialized error correction tools to the data before assembly. Many assemblers, on the other hand, do not require such preprocessing and handle sequencing errors during the assembly process itself. One of the challenges in genome assembly is to distinguish sequencing errors from true genomic variation.

1.3.4 Scaffolding

The final component of genome assembly is *scaffolding*, which takes place after construction of maximal length contigs. Scaffolding is a process through which contigs

are linked together into scaffolds in the order in which they appear in the genome, separated by gaps of known length. This is usually done based on information provided by specific sequencing technologies, indicating how reads are linked; this information is then transformed into linking of contigs. The task of scaffolding falls outside the scope of this thesis, but many available tools specialize in this—see e.g. [52] for a review.

1.4 Haplotype reconstruction

The copy-specific sequences of the genome of a polyploid organism are called *haplotypes*. These sequences generally differ in terms of the genetic variants affecting them: a position in the genome *(locus)* is called *homozygous* if the haplotypes show the same allele, and *heterozygous* otherwise. Reconstructing the individual haplotypes in an organism or population, also known as *haplotype-aware genome assembly*, is a difficult problem—in particular in a de novo setting. Beyond distinguishing between errors and true sequential variants, the true variants need to be assigned to the different genome copies.

Again, we can distinguish between reference-guided and de novo assembly when it comes to haplotype reconstruction. In reference-guided approaches to haplotype-aware genome assembly, variant calling leads to a list of loci showing variation and the observed alleles at each locus. The goal is then to assign variants to haplotypes, a process known as *phasing*. In de novo approaches, on the other hand, reads need to be linked together to form haplotype-specific contigs, also called *haplotigs*.

Haplotype-aware assemblies contribute to representations of all genomic content in a certain species or phylogenetic clade in the form of a *pan-genome* [74]. Such a structure can serve as a tool for joint analysis of the haplotypes from which it was built, but also as a catalog of known sequence variation, to be used as a reference for future analyses.

1.5 Applications

Haplotype-aware genome assembly plays an important role in many disciplines. In this thesis we focus mainly on viral quasispecies assembly, that is, the reconstruction of viral haplotypes within a single infection. In Chapter 3 we will touch upon haplotype assembly of the human genome, in particular the region coding for the major histocompatibility complex (MHC). Below, we discuss these and several other applications of haplotype-aware genome assembly.

1.5.1 Viral quasispecies

Viruses consist of small particles which cannot reproduce on their own. By infecting a host organism, they can use the replication apparatus of the host cells. Although viral genomes are relatively short compared to bacteria and eukaryotes, they are subject to very high mutation rates [35]. As the virus replicates rapidly during an infection, mutation and recombination lead to a variety of mutant strains. This ensemble of closely related viral strains populating the infected host is called a *viral quasispecies* [32]. The different mutant strains can show different phenotypic properties and appear at different frequencies within the population. Determining the haplotypes of the individual strains and their relative abundance rates can play a key role in assessing virulence and pathogenesis, as well as in therapy selection [39].

1.5.2 Major histocompatibility complex

The major histocompatibility complex is a highly polymorphic set of approximately 200 genes found in vertebrates, which are essential to the acquired immune system. The human MHC genes, also referred to as Human Leukocyte Antigens (HLA), are located in a region of 6 Mb on chromosome 6 of the human genome. Haplotype-aware reconstruction of the HLA genes and the entire MHC region plays an important role in disease association studies (in particular autoimmune diseases) and transplant rejection [24]. However, high variability within a population and high similarity between several genes make this assembly task particularly challenging.

1.5.3 Metagenomics

In metagenomics, sequencing technologies are used to characterize microbial systems. Metagenomic data sets consist of genetic material obtained from environmental samples, which usually contain a mixture of viral and bacterial genomes. Not only viruses, but also bacteria can exist as a population of closely related strains. The goal of metagenomics is to gain understanding of the ecology and evolution of microbial ecosystems. Examples of metagenomic studies include the analysis of microbes in ocean water [123] and sequencing of genetic material extracted from fecal samples to study the gut microbiome [44]. Haplotype-aware assembly of metagenomes from short-read data is extremely challenging due to the complexity and diversity of microbial communities, as well as the closeness of related strains and low relative abundances [107].

1.5.4 Transcriptomics

In transcriptomics, next-generation sequencing is applied to the complete set of RNA transcripts found in a cell. The different transcripts can be seen as individual haplo-

types, each of which may appear at a different frequency. The observed sequencing depths are indicative for the corresponding relative transcript abundance. The main challenge in RNA transcript assembly is to distinguish between spliced isoforms and similar transcripts within a gene family. Transcriptome assembly and subsequent comparison of assemblies across cells enable identification of genes that are differentially expressed between cell populations. This leads to further understanding of the chemical processes that take place in a cell, gene function annotation, and responses to different environments [42, 49].

1.5.5 Cancer genomics

Another situation where haplotype-aware genome assembly plays an important role is in analysis of tumor samples. Cancer cells replicate their genome and proliferate at much higher rates than healthy cells. This behaviour is driven by mutations, leading to a heterogeneous population of haplotypes within a single tumor. Analysis of these haplotype sequences allows for further understanding of tumor evolution, which may aid in development of effective cancer treatments [80]. NGS technologies enable low-cost sequencing of tumor populations at high sequencing depths. However, haplotype reconstruction remains a major challenge due to sequence heterogeneity, an unknown number of haplotypes, and complex patterns of variation.

1.6 Outline and contribution

The primary goal of the research presented in this thesis was to solve the viral quasispecies assembly problem. Until 2017, all approaches to this problem were reference-guided, introducing severe biases: highly divergent strains often got misassembled or were lost entirely. In order to get rid of these reference-related issues, we set out to find a de novo solution. As a result, we present the first de novo approach to successfully and efficiently reconstruct viral quasispecies at full length. Inspired by this success, we also present a de novo assembly algorithm for genomes of known ploidy.

Our contribution was made in four steps, each presented as a chapter in this thesis. These chapters are based on research articles that have been published in or submitted to a scientific journal in the same form as they appear here. Although **Chapters 3 and 4** follow up on ideas presented in **Chapter 2**, and **Chapter 5** presents alternatives to the method described in **Chapter 4**, each chapter is self-contained and can be read in isolation.

Chapter 2 presents SAVAGE (Strain Aware VirAl Genome assEmbler), a computational tool for viral quasispecies reconstruction without the need for a high-quality reference genome. SAVAGE makes use of either FM-index-based data structures or

ad-hoc consensus reference sequence for constructing overlap graphs from patient sample data. Following an iterative scheme, a new overlap assembly algorithm then efficiently reconstructs haplotigs from this overlap graph. In benchmark experiments on simulated and real deep coverage data, SAVAGE drastically outperforms generic *de novo* assemblers as well as specialized viral quasispecies assemblers in terms of error rates.

Chapter 3 presents POLYTE (POLYploid genome fitTEr) as a new approach to de novo generation of haplotigs for diploid and polyploid genomes of known ploidy, inspired by the success of overlap graph-based assembly in **Chapter 2**. The main difference between POLYTE and SAVAGE is the type of data that is targeted: while viral quasispecies are typically sequenced at ultra-deep coverage (10.000–100.000x), this chapter focuses on data sets of low to medium coverage values (10–100x). POLYTE adopts ideas from **Chapter 2**, following an iterative overlap graph-based scheme where in each iteration reads or contigs are joined, based on their interplay in terms of an underlying haplotype-aware overlap graph. With each iteration, contigs grow while preserving their haplotype identity.

In order to deal with low coverage sequencing data, edge constraints for the overlap graph are less restrictive in comparison to the previous chapter. This, however, increases the number of spurious edges and thereby the risk of assembling false haplotypes. We minimize this risk by developing a procedure to reduce the number of spurious edges in the overlap graph. Experiments on both real and simulated data demonstrate that POLYTE establishes new standards in terms of error-free reconstruction of haplotype-specific sequences. As a consequence, POLYTE outperforms state-of-the-art approaches in various relevant aspects, where advantages become particularly distinct in polyploid settings.

Chapters 2 and 3 illustrate the benefits of reference-genome-independent (*de novo*) approaches over reference-guided approaches, where reference-induced biases can become overwhelming. Especially when dealing with highly divergent genomes, reference-guided methods are unable to reconstruct individual haplotypes at low error rates. De novo methods, on the other hand, yield highly accurate, yet rather short contigs. The remaining challenge is to reconstruct full-length haplotypes together with their abundances from these contigs; this challenge is addressed in **Chapters 4 and 5**.

In **Chapter 4** a de novo approach to extend pre-assembled contigs into viral haplotypes based on variation graphs is presented: Virus-VG. This method constructs a variation graph from the short input contigs, without making use of a reference genome or any other prior information. Then, we enumerate all maximal-length paths through this graph that maximally concatenate the contig subpaths. To obtain a selection of paths that reflects the haplotypes present in the sample, a minimization problem is solved, yielding a selection of maximal-length paths that is optimal in terms of being

compatible with the read coverages computed for the nodes of the variation graph. The resulting selection of paths is output as the assembled haplotypes, together with their abundances. Benchmark experiments show significant improvements in assembly contiguity compared to the input contigs, while maintaining low error rates compared to the state-of-the-art viral quasispecies assemblers.

An immediate limitation of the Virus-VG algorithm is that it is an exponential time algorithm due to the path enumeration step. Although experiments in **Chapter 4** show that this approach suffices to solve the practical problems under consideration, it does not scale well to larger genomes: the path enumeration step would simply explode.

Chapter 5 shows how we can avoid enumerating all possible paths by an appropriate flow formulation of the problem and presents VG-flow, a computational tool that implements these new ideas. We cast the assembly problem into a min-cost flow optimization problem, yielding abundance estimates for each of the input contigs. This optimization problem can be solved in polynomial time and avoids the costly path enumeration step used in Virus-VG. Based on the computed contig abundances, we greedily select a collection of candidate haplotypes, which are subsequently used as input for the linear program described in the previous chapter. Thus, this chapter describes an efficient solution to the quasispecies reconstruction problem from pre-assembled contigs.

Together, the approaches of **Chapters 2, 4, and 5** present a complete solution to the viral quasispecies assembly problem. In addition, the computational machinery described in **Chapters 3 and 5** contributes to haplotype-aware genome assembly of polyploid species other than viruses and has the potential to take a big step ahead in haplotype-aware genome assembly in general. In **Chapter 6**, we further discuss the implications of this work and provide perspectives on future research in the field of haplotype-aware genome assembly.

1.7 Final remarks

There is much more to tell about the design, implementation, and benchmarking of the methods presented in this thesis than what is written in Chapters 2-5. To preserve the textual flow and to avoid burdening the reader with too much information, some algorithmic details and experimental results are presented in supplementary material available from the publishers' websites.

Chapters 2, 4 and 5 all present assembly results for SAVAGE. However, the careful reader may note the assembly quality has improved in later chapters compared to Chapter 2. Indeed, since its first version in 2016, the SAVAGE algorithm has been under continuous development, leading to improvements in assembly quality. The results presented in Chapter 2 are based on SAVAGE version 0.1.0, while results in Chapters 4

and 5 are based on SAVAGE version 0.4.0. The corresponding changes to the algorithm are described in the SAVAGE changelog[1]. Further note that in Chapters 4 and 5, when we refer to SAVAGE we always imply the index-based de novo algorithm, introduced in Chapter 2 as SAVAGE-de-novo.

[1] https://github.com/HaploConduct/HaploConduct/blob/master/savage/CHANGELOG.md

CHAPTER 2

DE NOVO ASSEMBLY OF VIRAL QUASISPECIES USING OVERLAP GRAPHS

A viral quasispecies, the ensemble of viral strains populating an infected person, can be highly diverse. For optimal assessment of virulence, pathogenesis and therapy selection, determining the haplotypes of the individual strains can play a key role. As many viruses are subject to high mutation and recombination rates, high-quality reference genomes are often not available at the time of a new disease outbreak. In this chapter we take the first steps towards de novo haplotype reconstruction in viral quasispecies.

We present SAVAGE, a computational tool for reconstructing individual haplotypes of intra-host virus strains without the need for a high-quality reference genome. We show that overlap graph-based viral quasispecies assembly is feasible and allows for the construction of high quality, strain-specific contigs. In benchmark experiments on both simulated and real deep coverage data sets, SAVAGE drastically outperforms generic *de novo* assemblers as well as specialized viral quasispecies assemblers. We also apply SAVAGE on two deep coverage samples of patients infected by Zika and hepatitis C virus, respectively, which sheds light on the genetic structures of the respective viral quasispecies.

Published as:
J.A. Baaijens, A. Zine El Aabidine, E. Rivals, and A. Schönhuth. De novo assembly of viral quasispecies using overlap graphs. *Genome research*, 27(5): 835–848, 2017.

Supplementary material: http://www.genome.org/cgi/doi/10.1101/gr.215038.116.

2.1 Introduction

Viruses such as HIV, the Zika and the Ebola virus, populate their hosts as an ensemble of genetically related but different mutant strains, commonly referred to as *viral quasispecies*. These strains, each characterized by its own haplotypic sequence, are subject to high mutation and recombination rates [32, 34]. Sequencing methods aim at capturing the genetic diversity of viral quasispecies present in infected samples; the promise is that next-generation sequencing (NGS) based methods will assist clinicians in selecting treatment options and other clinically relevant decisions.

Ideally, a *viral quasispecies assembly* characterizes the genetic diversity of an infection by presenting all of the viral haplotypes, together with their abundance rates. There are two major challenges in this.

(1) The number of different strains is usually unknown. Furthermore, two different strains can differ by only minor amounts of distinguishing mutations. Last but not least, abundance rates can be as low as the sequencing error rates, which hampers the detection of true mutations present at low frequency.

(2) Due to the great diversity and the high mutation rates, reference genomes representing high-quality consensus genome sequences can be obsolete at the time of the disease outbreak. The lack of a suitable reference genome is a major hindrance for many viral quasispecies assembly approaches.

It is important to understand that all existing assembly methods fail to address either the first or the second point. Recent *reference-guided approaches* specialized in viral quasispecies assembly suggested statistical frameworks modelling the driving forces underlying the evolution of viral quasispecies. While previous approaches focused mostly on local reconstruction of haplotypes [50, 99, 130, 131], more advanced approaches aimed at global reconstruction of haplotypes, for example, by making use of Dirichlet process mixture models [97], hidden Markov models [119], or sampling schemes [98]. There are also recent combinatorial approaches which compute paths in overlap graphs [6], enumerate maximal cliques in overlap graphs [118], or compute maximal independent sets in conflict graphs [73]. While these approaches soundly address point (1), the vast majority of them depends on high-quality reference sequence as a backbone to their methods, which in turn is the reason why they fail to address (2). Hence, when confronted with hitherto unknown, significantly deviating mutation patterns, these approaches fail to perform sufficiently well.

On the other hand, *de novo assembly approaches* do not depend on reference genomes. Although there exist numerous *de novo* approaches for mammalian genome assembly, – see e.g. [17, 47, 105] for comparative evaluations – these generic methods are not well suited for the viral quasispecies assembly problem. The key difference is that mutation rates in viruses are orders of magnitude higher than in eukaryotes, result-

ing in multiple polymorphic sites within a single read [32, 34]. This makes it possible to phase mutations into separate haplotypes; however, generic assembly approaches do not exploit this property. Rather, generic assemblers aim at reconstructing one single consensus sequence or are not designed to handle genomes of heavily polyploid organisms. In this regard, note that there are *de novo* assemblers that specialize in viral genome assembly already [51, 129]. However, also these specialized approaches aim at assembling consensus genomes rather than strain-specific sequence, where the goal is to construct new reference rather than individual sequence. To our knowledge, the only existing *de novo* approach for haplotype-resolved viral quasispecies assembly is MLEHaplo [72]. As a consequence, while addressing (2), most existing de novo assembly methods fail to address point (1) to a satisfactory degree.

A possible principled issue is that nearly all of the NGS based genome assemblers, including the above-mentioned specialized *de novo* viral quasispecies approaches, rely on the *de Bruijn graph* as assembly paradigm. Thereby, reads are decomposed into k-mers, where k is usually considerably smaller than the read length. As a generalization of this concept the paired de Bruijn graph has been introduced [78], which incorporates mate pair information into the graph structure itself instead of analyzing mate pairs in a post-processing step, which yields larger contigs in the assembly. As mentioned above, it is imperative in viral quasispecies assembly to distinguish low-frequency mutations from sequencing errors. While low-frequency mutations are genetically linked, hence co-occur within different reads, sequencing errors do not exhibit patterns of co-occurrence. The detection of patterns of co-occurrence is decisively supported by examining reads at their full length, but this information cannot be exploited with de Bruijn graphs. Overlap graphs on the other hand make use of full-length reads and do not decompose them into smaller parts; hence, we reason that the overlap graph paradigm suits the problem of viral quasispecies assembly better.

The only existing method for viral quasispecies assembly based on overlap graphs is HaploClique [118]. Although this method is reference guided, it uses the reference solely for providing anchor points for constructing an overlap graph. Unlike in many other approaches [30, 119, 130, 131], the haplotype sequences are then assembled from the reads, and not from the reference. While providing inspiration in general, the HaploClique algorithm has proven to require excessive computational resources already on data sets of relatively low coverage (1000x and more). The reason is that it is based on the enumeration of maximal cliques, which is exponential in the read coverage, both in terms of runtime and space. We therefore present a novel, more efficient algorithm for the clique enumeration part of the assembly algorithm.

There are two exit strategies to resolve the issue of the possible lack of a reference genome. The *first strategy* is to construct consensus genome sequence from the patient samples themselves, using one of the available *de novo* consensus genome assemblers

(among which, the most popular tool is VICUNA [129]), and to subsequently run one of the reference-guided approaches using this ad-hoc consensus as a reference. This strategy has also been suggested by [73] and we shall further explore it here. The *second strategy* is to construct an overlap graph directly from the patient sample reads. Subsequently, we employ a ploidy-aware assembly algorithm that can extract strain-specific sequences from overlap graphs. The challenge is that constructing overlap graphs requires a pairwise comparison of all reads, which, for deep coverage data sets, requires sophisticated indexing techniques to be feasible. Here, we show how to make efficient use of FM-index based techniques [121] to construct overlap graphs without any need for a reference genome. As such, we provide *the first approach for de novo assembly of viral quasispecies based on overlap graphs.*

In summary, we make relevant contributions for

(i) the construction of overlap graphs from deep coverage read data and

(ii) viral quasispecies assembly using the overlap graph assembly paradigm.

In combination, we present SAVAGE (Strain Aware VirAl GEnome assembly), a method that allows for reference-free assembly of viral quasispecies from sequencing data sets of deep coverage (20 000x and more). In this, we do not only provide the first genuine *de novo* viral quasispecies assembly approach based on overlap graphs, but we also provide the first method that can exploit ad-hoc consensus sequence generated from patient samples, as computed for example by VICUNA [129], for high-performance viral quasispecies assembly.

2.2 Results

We have designed and implemented SAVAGE (Strain Aware VirAl GEnome assembly), a method for *de novo* viral quasispecies assembly based on overlap graphs. In this section, we provide a high-level description of the algorithmic approach and analyze its performance, also in comparison to state-of-the-art viral quasispecies assembly tools and several established generic genome assemblers. Finally, we present assembly results using SAVAGE on two real virus samples from patients infected by the Zika virus and hepatitis C virus, respectively. We refer to the Methods section for any methodological details.

2.2.1 Approach

Our algorithm proceeds in three stages (panel A of Figure 2.1), each of which iteratively clusters the input sequences and extends them to unique haplotypes. While *Stage a* has the original reads as input and contigs as output, *Stage b* has these contigs as input and maximally extended contigs as output. The extended contigs are supposed to reflect in-

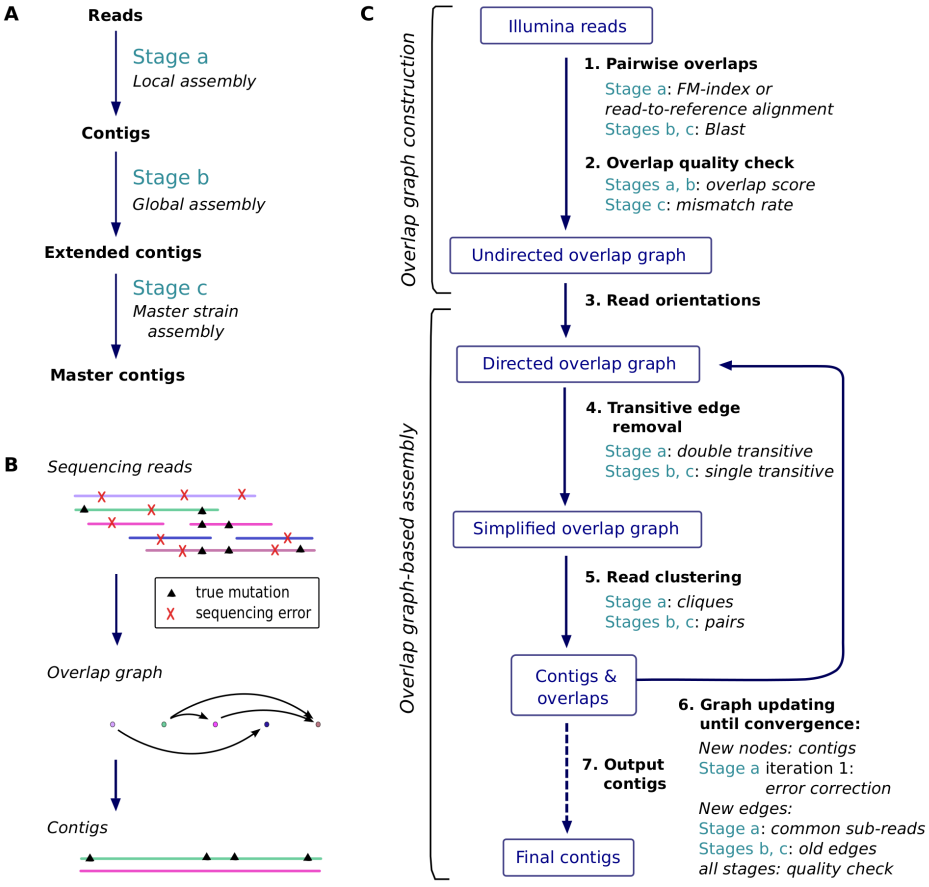

Figure 2.1: An overview of the workflow and algorithms of SAVAGE. **A.** The three stages of SAVAGE. Each assembles sequences into longer sequences. For clarity, we assign different names to the sequences output by each stage: contigs, maximally extended contigs, and master contigs, respectively. **B.** Principle of overlap graph construction and distinction among the reads between errors and shared mutations. **C.** Each stage has two steps: first, the overlap graph construction, second, assembly. This panel summarizes the differences in each step between the three stages. During overlap graph-based assembly, steps 4 to 6 are repeated iteratively until there are no edges left in the overlap graph.

dividual haplotype sequences. Finally, the optional *Stage c* merges maximally extended contigs into master contigs, each representing a group of very closely related strains. This reflects the existence of **master strains** in many viruses, where each individual haplotype deviates from one of the master strains by only a relatively minor amount of mutations (the ensemble of which is commonly referred to as *mutant class* in the literature and reflects a viral subpopulation—see e.g. [32]). Each stage is divided into **overlap graph construction** (upper part of panel C in Figure 2.1) and **overlap graph based assembly** (lower part of panel C in Figure 2.1). Between the stages, this generic structure only differs in the details.

The strength of overlap graphs for viral quasispecies assembly is in identifying co-occurring mutations, thus enabling the phasing of mutations from the same strain. We distinguish sequencing errors from true mutations by posing very strong constraints on the overlaps in terms of minimal overlap length and sequence similarity. In addition, we make use of paired-end read information. This results in a very conservative overlap graph, where an edge indicates that two sequences are very likely to originate from the same virus strain. Therefore, by enumerating cliques in the overlap graph we cluster the reads per strain, thus reconstructing the individual haplotypes of the viral quasispecies.

We construct overlap graphs in two steps: first, pairs of reads are determined that share sufficiently long and well-matching overlaps, followed by a statistical evaluation of the quality of each overlap. We explore two options for finding all such overlap candidates. The first option is to apply a completely *de novo* procedure using FM-index based techniques [121]. The second option is to align all reads against a reference genome, such that read-to-read alignments can be induced from the read-to-reference alignments. However, in case of a viral outbreak there may not be a suitable reference genome available; we target such cases by constructing an ad-hoc consensus sequence from the patient samples, as computed by VICUNA [129].

SAVAGE offers three different modes, corresponding to the different approaches to overlap graph construction described above: **SAVAGE-de-novo** uses the first option and is therefore completely reference-free, while **SAVAGE-b-ref** uses the second option and thus relies on a bootstrap reference sequence. For benchmarking purposes we also consider **SAVAGE-h-ref**, which takes as input an existing, high quality reference sequence.

2.2.2 Benchmark data

For benchmark experiments and performance analysis, we considered several simulated data sets, one gold standard benchmark from real sequencing reads, and two real patient samples. For the simulated data sets, sequencing reads were created using the simulation software SimSeq (see Methods).

Simulated benchmarks. We created five simulated data sets for benchmarking, consisting of 2 × 250 bp Illumina MiSeq reads and representing quasispecies infections from different viruses: human immunodeficiency virus (HIV), hepatitis C virus (HCV), and Zika virus (ZIKV). We varied the number of strains per sample, as well as the relative abundances of those strains and the pairwise divergence between strains. To get data sets as realistic as possible, we used true viral genomes from the NCBI database and Illumina MiSeq error profiles during simulations. Characteristics of each benchmark are given in Table 2.1 and additional information can be found in Supplementary Methods.

Lab mix. In addition to the simulated benchmarks, we also considered a real Illumina MiSeq (2 × 250 bp) data set with an average coverage of ~20 000x, obtained from a lab mixture of five HIV strains (see also Table 2.1). This data set was recently presented as a gold standard benchmark [30] and is available at https://github.com/cbg-ethz/5-virus-mix; we will refer to it as the *lab mix*.

Divergence-vs-ratio. To analyze the combined effect of the levels of divergence and of the relative abundance of the strains, we constructed 36 additional data sets as follows. Starting from the HIV-1 89.6 haplotype, we created six alternative haplotypes by introducing, respectively, 0.5%, 0.75%, 1%, 2.5%, 5%, and 10% random mutations. For each of those six alternative strains, we created six data sets by simulating reads (2 × 250 bp Illumina MiSeq) from the mutated strain and the original at a ratio of 1:1, 1:2, 1:5, 1:10, 1:50, and 1:100, respectively, with a total coverage of 500x per data set.

Zika virus sample. We applied SAVAGE to a sample of Asian-lineage Zika virus (ZIKV) consisting of Illumina MiSeq 2x300 bp sequencing reads (~30,000x coverage) obtained from a rhesus macaque after four days of infection [33, animal 393422]. This data set is available in the NCBI Sequence Read Archive under experiment SRX1678783, run SRR3332513.

Hepatitis C virus sample. In addition to the Zika virus sample, we also used a hepatitis C virus (HCV) sample of approximately 80 000x coverage, covering a region of ~3000 bp containing the HS5B gene. This data set is available in the NCBI Sequence Read Archive under experiment SRX396803, run SRR1056035.

2.2.3 Evaluation preliminaries

In case of a viral outbreak, the agent and its genome may be unknown (or may have significantly diverged from closely related strains such that available reference sequences are potentially inadequate for analysis), and the samples taken from infected patients contain an unknown number of divergent strains. Here, we target these cases where no reference genome is available. A sample sequenced with next-generation sequencing technology delivers enough reads and sufficient coverage to allow a *de novo* assembly of a viral genome (here, we mean a single genome assembly, not a quasispecies assembly),

Data set	Virus type	Genome size (bp)	Average coverage	Strain count	Strain abundance	Pairwise divergence
600x HIV mix	HIV-1	9478–9719	600x	5	20 %	1–6 %
5-strain HIV mix	HIV-1	9478–9719	20 000x	5	5–28 %	1–6 %
10-strain HCV mix	HCV-1a	9273–9311	20 000x	10	5–19 %	6–9 %
3-strain ZIKV mix	ZIKV	10251–10269	20 000x	3	16–60 %	3–10 %
15-strain ZIKV mix	ZIKV	10251–10269	20 000x	15	2–13 %	1–10 %
Lab mix	HIV-1	9478–9719	20 000x	5	10–30 %	1–6 %

Table 2.1: Characteristics of benchmark data sets. For each benchmark we specify virus type, genome size, average coverage, strain count, relative abundance, and pairwise divergence. For the 600x HIV mix, the strains were homogeneously distributed with a relative abundance of 20% each.

to be used as an ad-hoc reference genome for further analyses. However, such genome sequences may not represent any of the true viral haplotypes present in the sample sufficiently well.

In the remainder of this paper, all assembly algorithms were run using default settings. Evaluations of assemblies were performed with MetaQUAST [82], which computes the usual statistics – number of contigs, largest contigs, N50, misassembled contig length, target genome(s) covered, and error rates – and we accounted only for contigs larger than a threshold of 500 bp. A contig is called misassembled if it contains at least one misassembly, i.e., a position where the left and right flanking sequences align to the true genomes with a gap or overlap of more than 1 kbp, or align to different strands, or even align to different strains.

We compare *de novo* methods and reference-guided approaches. While *de novo* algorithms proceed by iteratively extending contigs until some convergence criterion is met, reference-guided approaches alter the reference sequence until a set of haplotypes is obtained that is supposed to represent the quasispecies. By altering the reference genome, all output sequences have the same length, which means that the N50 score equals the length of the output sequences. For *de novo* approaches, on the other hand, the N50 score provides an indication of the contig length distribution.

2.2.4 Failure of existing de novo assemblers on low-frequency strains

We explored the ability of generic genome assemblers to reconstruct a viral quasispecies. From the broad collection of tools available, we selected four assemblers: SGA [110], SOAPdenovo2 [69], SPAdes [9], and metaSPAdes [89]. The first two methods, SGA and SOAPdenovo2, are generic assemblers, mostly used on mammalian genomes. SPAdes was originally designed for bacterial genomes, and metaSPAdes is a version of SPAdes

adapted for metagenome assembly.

First, we evaluate performance on all simulated benchmarks. Table 2.2 presents results for all methods on the 5-strain HIV mix, the 10-strain HCV mix, and the 15-strain ZIKV mix. The only method capable of assembling at least half a viral quasispecies on a 20 000x simulated data set is SPAdes, the only close alternative being metaSPAdes with 45.9% on the 10-strain HCV mix. For the 5-strain HIV mix and the 10-strain HCV mix, SPAdes assembles 91.3–91.7% (SAVAGE-de-novo: ≥ 99.6%) of the true viral genomes at an error rate of 0.015 – 0.084% (SAVAGE-de-novo: 0.004%), showing that SPAdes misses to assemble a considerable fraction of the quasispecies. This becomes more evident on the 15-strain ZIKV mix, which contains several low-frequency strains: SPAdes only recovers 65.6% of the target genomes (SAVAGE-de-novo: 99.4%). The explanation for this is that SPAdes misses to assemble strains of low frequency, as Figure 2.2 further reveals: here, a comparison of all approaches is shown when at most a bootstrap reference is provided. The performance of each approach is evaluated on each of the strains of the 20 000x benchmarks from Table 2.1 individually, and results are stratified by the relative abundances of the strains. We see that SPAdes recovers only 46.8% of the strains of frequency of less than 5%.

Similar results for the 600x HIV mix and the 3-strain ZIKV mix can be found in Supplementary Tables S1 and S2; these are relatively easy data sets, since neither contains any low-frequency strains. Both SOAPdenovo2 and SPAdes perform reasonably on the 600x data set, reconstructing 78.9% and 87.8% of the viral quasispecies, respectively. SGA and metaSPAdes, on the other hand, do not recover more than 19% of the quasispecies. For the 3-strain ZIKV mix, only SPAdes is able to reconstruct more than 40% of the quasispecies; in fact, it finds 99.6% of the target genomes, performing almost perfectly on this low-ploidy data set, which is no surprise because assemblers like SPAdes generally target at genomes of limited ploidy.

Finally, we consider the lab mix, which is based on real data and hence the most challenging benchmark. Table 2.3 presents results for all methods. SGA, SOAPdenovo2, SPAdes, and metaSPAdes all perform quite similarly, reconstructing only 41.0–53.7% of the viral quasispecies at very high error rates (1.1–2.0%). This shows that each of these assemblers has difficulty distinguishing sequencing errors from true variants, thus pointing out the need for specialized viral quasispecies assemblers.

The first specialized *de novo* assembler is now available [72]. We ran this method, called MLEHaplo, on our benchmark data sets. Unfortunately, it could only handle the 600x HIV mix; for all 20 000x benchmarks, MLEHaplo did not finish within a week and used more than 140GB of main memory per data set. On the 600x HIV mix, it performed very poorly, reconstructing only 10% of the target genomes at a mismatch rate of more than 2%.

2.2.5 Dependence of reference based approaches on reference genome quality

Reference based quasispecies assembly tools proved to perform adequately when a high quality reference genome is available [97, 130]. We question whether reference based approaches could yield appropriate quasispecies assemblies if provided with a *de novo* assembled genome sequence obtained from the sample reads, rather than a high quality reference genome. To address this point, we compared state-of-the-art methods PredictHaplo [97] and ShoRAH [130] on our benchmarks (Table 2.1) in two settings: either with a high quality reference genome, or with a genome sequence obtained by running the VICUNA assembler [129] on the sample reads. We refer to the former as a **high quality reference genome**, denoted *h-ref*, and the latter as a **bootstrap reference genome**, denoted *b-ref*. The quality of the output assemblies, as evaluated with MetaQUAST, is described in Tables 2.2 and 2.3, as well as Supplementary Tables S1 and S2.

For PredictHaplo and ShoRAH, the number of output sequences provides an estimate of the total number of strains in the quasispecies, since each output sequence represents a putative strain in the quasispecies. In Table 2.2, we see that on all benchmarks except the 15-strain ZIKV mix, the number of output sequences for PredictHaplo is very close to the true number of strains. For the 3-strain ZIKV mix, both the high quality reference genome and the bootstrap reference genome lead to a perfect assembly of 3 sequences without any mismatches and less than 0.042% indels (Supplementary Table S2). But considering the remaining (more challenging) data sets, we see that using a bootstrap reference genome causes a serious loss in the fraction of target genomes recovered by PredictHaplo (compared to using a high quality reference). On the 600x HIV mix and the lab mix, using the bootstrap reference even results in 100% of the sequences being misassembled (Supplementary Table S1). Only for the 15-strain ZIKV mix the difference between the h-ref and b-ref approaches is small: both recover only 53% of the target genomes (8 out of 15 strains – see Table 2.2).

For ShoRAH, we observe that for all data sets the number of output sequences is one or two orders of magnitude larger than the true number of strains. In addition, the mismatch rate is high compared to other methods, varying between 2.4% and 4.4% on the simulated 20 000x benchmarks. Unfortunately, we can only compare the bootstrap reference and high quality reference approaches on the HIV data, because ShoRAH-h-ref crashed repeatedly on the HCV and ZIKV benchmarks. Remarkably, the bootstrap reference approach increases target genome coverage from 39.4% to 93.8% on the 5-strain HIV mix (Table 2.2). However, in both the 20 000x HIV mix and the 600x HIV mix we see that the bootstrap reference also results in a small fraction of the total sequence length being misassembled (7.0% and 1.6%, respectively). This effect becomes

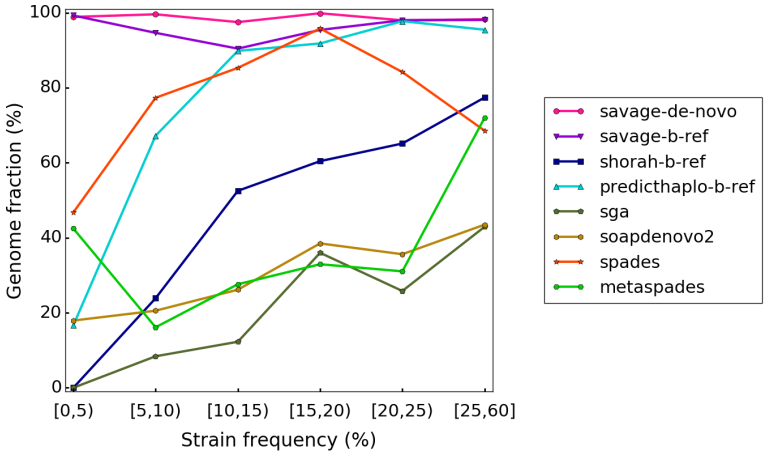

Figure 2.2: Target genome fraction recovered per strain for all 20 000x benchmarks, stratified by strain frequency.

much more apparent on the lab mix, with 89.3% of the total sequence length being misassembled. This shows that, similar to PredictHaplo, the quality of the ShoRAH assembly is highly dependent on that of the reference genome sequence.

Both tools, especially PredictHaplo, seem valuable when the reference genome is closely related to sample strains, but inadequate to handle cases where a good reference genome is unavailable. Moreover, Figure 2.2 shows that both PredictHaplo and ShoRAH have trouble reconstructing low-frequency strains, recovering less than 17% of the low-frequency (<5%) target strains. These results emphasize the need for new assembly approaches that are independent of a reference genome.

2.2.6 SAVAGE evaluation

For the sake of comparison, we ran SAVAGE on the same benchmarks as above (Table 2.1) in both *de novo* mode and reference mode, both with default parameters. The 20 000x coverage data sets were split into patches of 750x each, on which we applied SAVAGE *Stage a*. Subsequently, all *Stage a* contigs were put together into one big collection of contigs and used as input for *Stage b* (Supplementary Fig 1).

Table 2.2 presents the evaluation results on simulated benchmarks of the *Stage b* maximally extended contigs for each of the three modes: SAVAGE-h-ref with a high quality reference genome, SAVAGE-b-ref with the genome assembled by VICUNA, and SAVAGE-de-novo (without reference). Remember that all *de novo* assemblers, including SAVAGE, proceed by progressively assembling longer and longer contigs starting

	# contigs ≥ 500 bp	largest contig	N50	MAC length (%)	target genomes (%)	N-rate (%)	mismatches (%)	indels (%)
5-strain HIV mix								
PredictHaplo-h-ref	5	9720	9720	0	99.6	0.603	0.085	0.102
PredictHaplo-b-ref	5	9578	9578	0	93.8	0.284	0.110	0.104
ShoRAH-h-ref	289	9514	9514	0	39.4	0.268	2.403	0.016
ShoRAH-b-ref	242	9501	9501	7.0	93.8	0.127	3.197	0.124
SAVAGE-de-novo	36	9413	4913	0	99.8	0	0.004	0
SAVAGE-h-ref	28	9634	5027	0	99.6	0	0.004	0
SAVAGE-b-ref	59	9463	2424	0	99.5	0.002	0.071	0.002
SGA	36	1034	650	0	32.4	0	1.294	0.026
SOAPdenovo2	36	844	516	0	35.7	0	0.633	0
SPAdes	14	9789	5873	0	91.7	0	0.084	0.002
metaSPAdes	13	7044	5159	0	32.7	0	1.681	0.013
10-strain HCV mix								
PredictHaplo-h-ref	9	9313	9313	0	90.0	0.004	0.402	0.010
PredictHaplo-b-ref	9	7636	7636	0	73.8	0.006	0.053	0
ShoRAH-h-ref	-	-	-	-	-	-	-	-
ShoRAH-b-ref	639	7570	7570	0	56.9	0	4.381	0.011
SAVAGE-de-novo	46	9297	8248	0	99.6	0.002	0.004	0
SAVAGE-h-ref	85	9247	3716	0	99.6	0	0.004	0
SAVAGE-b-ref	84	7802	2943	0	86.0	0	0.001	0
SGA	33	832	638	0	18.1	0	1.439	0
SOAPdenovo2	41	926	531	0	22.0	0	0.551	0
SPAdes	13	9311	8582	0	91.3	0	0.015	0
metaSPAdes	81	3041	1549	0	45.9	0	2.133	0
15-strain ZIKV mix								
PredictHaplo-h-ref	8	10258	10258	0	53.3	0.032	0.147	0.046
PredictHaplo-b-ref	8	10270	10270	0	53.3	0.001	0.121	0.004
ShoRAH-h-ref	-	-	-	-	-	-	-	-
ShoRAH-b-ref	493	10117	10117	0	26.3	0.053	4.403	0.017
SAVAGE-de-novo	607	9282	2103	0	99.4	0.002	0.016	0
SAVAGE-h-ref	641	10243	1935	0	99.4	0.002	0.006	0
SAVAGE-b-ref	604	9079	2018	0	99.5	0.002	0.011	0
SGA	0	-	-	-	0	-	-	-
SOAPdenovo2	56	1025	562	0	21.0	0	0.545	0
SPAdes	60	10269	2577	0	65.6	0	0.131	0
metaSPAdes	37	6495	3926	0	17.5	0	1.200	0

Table 2.2: Assembly results per method on simulated HIV, HCV, and ZIKV benchmarks (20 000x coverage).

	# contigs ≥ 500 bp	largest contig	N50	MAC length (%)	target genomes (%)	N-rate (%)	mismatches (%)	indels (%)
PredictHaplo-h-ref	5	9642	9642	0	99.2	0.259	0.615	0.104
PredictHaplo-b-ref	5	11000	11000	100	94.5	0.425	0.011	0.136
ShoRAH-h-ref	160	9581	9581	0	98.9	0.378	3.203	0.113
ShoRAH-b-ref	169	10854	10854	89.3	99.0	0.770	0.911	0.165
SAVAGE-de-novo	846	1221	588	0.1	92.6	0.183	0.161	0.040
SAVAGE-h-ref	848	1167	588	0.3	91.5	0.220	0.251	0.036
SAVAGE-b-ref	828	1226	595	0.1	92.2	0.162	0.101	0.040
SGA	60	1117	635	1.5	41.0	0	1.811	0.046
SOAPdenovo2	56	984	591	1.5	41.9	0	1.655	0.114
SPAdes	60	2952	591	1.2	42.6	0	1.154	0.097
metaSPAdes	27	4543	3266	0	53.7	0	2.045	0.100

Table 2.3: Assembly results per method on the HIV lab mix, a gold standard benchmark containing real sequencing data (20 000x coverage).

from the raw reads, until finally, each output contig may (partially) cover the target genomes. Hence, unlike for PredictHaplo and ShoRAH, the number of contigs cannot be interpreted directly as a number of strains.

With a reference, the results of SAVAGE-h-ref and SAVAGE-b-ref are very similar: the contigs cover more than 99% of the target genomes, with the largest contig length close to the genome size of the virus in question. The mismatch, indel, and N rates are globally better than those offered by PredictHaplo and ShoRAH: the indel and N rates are respectively one or two orders of magnitude lower. Above all, the contigs are free of misassemblies (MAC length is 0%). Strikingly, providing a high quality reference genome or a bootstrap genome makes little difference, and on some data sets SAVAGE with a bootstrap genome achieves better results for certain statistics (higher N50, larger target genome fraction, lower mismatch rate for the 15-strain ZIKV mix in Table 2.2). These observations also hold on the lab mix (Table 2.3), where SAVAGE-ref recovers 91.5–92.2% of the target genomes at a mismatch rate of 0.101–0.251% and very low indel rates.

On all benchmarks, SAVAGE-de-novo delivers an assembly that is qualitatively at least as good as the SAVAGE-h-ref and -b-ref assemblies. Figure 2.2 shows that, in terms of target genome recovered, SAVAGE-de-novo slightly but consistently outperforms SAVAGE-ref. More importantly, this figure shows that both SAVAGE-de-novo and SAVAGE-b-ref greatly outperform *all* other methods, especially on low-frequency strains (i.e., frequency < 10%).

To analyze the effect of read length on SAVAGE assembly performance, we also built

a 5-strain HIV mix with the exact same properties as given in Table 2.1, but with shorter reads (2x150 bp). We evaluated the resulting maximally extended contigs for SAVAGE-de-novo, SAVAGE-h-ref, and SAVAGE-b-ref (Supplementary Table 3). Compared to the original 5-strain HIV mix, which has 2×250 bp reads, SAVAGE produces a more fragmented assembly but still covers 90.6–98.4% of the target genomes with mismatch rates between 0 and 0.006%.

Overall, SAVAGE can process samples containing a mixture of multiple strains and recover most of the target genomes with a high level of sequence quality. It performs slightly better in *de novo* mode than with a reference sequence and also performs well on shorter sequencing reads. Moreover, compared to existing methods, our approach does not suffer from misassemblies. For SAVAGE-de-novo, the misassembled contig (MAC) length is 0% on all simulated data sets and 0.1% on the lab mix, which drastically outperforms all approaches that reach ≥ 90% genome coverage and operate *without a high quality reference*. Moreover, SAVAGE can take advantage of a bootstrap reference sequence built by a single genome assembler. Finally, SAVAGE offers contigs with improved mismatch and indel rates, especially on low-frequency strains.

2.2.7 Runtime and memory usage

We evaluate algorithm efficiency on both the 600x and the 20 000x simulated HIV mix, as well as the lab mix. We report CPU time and maximum memory usage for all methods evaluated previously on each of these HIV data sets in Supplementary Table 4. In terms of CPU time, SAVAGE-b-ref was considerably faster than SAVAGE-de-novo, with 6.4 versus 19 minutes on the 600x HIV mix, 449 versus 5296 minutes on the 20 000x HIV mix, and 850 versus 7495 minutes on the *lab mix*. This was to be expected, since *de novo* overlap graph construction requires enumeration of all approximate suffix-prefix overlaps among the reads. In comparison, PredictHaplo was faster but of the same order of magnitude as SAVAGE-b-ref with 7, 223, and 158 minutes, respectively. ShoRAH was comparable to SAVAGE and PredictHaplo on the 600x HIV mix (12 min) but very slow on the 20 000x data (22256–32375 min). The de Bruijn graph-based assemblers (SOAPdenovo2, SPAdes, and metaSPAdes) were very fast on all data sets, with a CPU time of 0.15–2 minutes on the 600x HIV mix, 5–46 minutes on the 20 000x HIV mix, and 6–166 minutes on the lab mix. The generic assembler SGA was considerably slower, with 24, 164, and 300 minutes, respectively. Finally, with 54 minutes on the 600x data MLEHaplo was the slowest, which also points out why it could not finish the 20 000x benchmarks.

Peak memory usage varied between 0.04 GB (PredictHaplo) and 8.4 GB (SPAdes, metaSPAdes) for the 600x HIV mix, between 0.5 GB (SGA) and 10 GB (ShoRAH) for the 20 000x HIV mix, and between 0.7 GB (SGA) and 12 GB (ShoRAH) for the lab mix. Both

SAVAGE-de-novo and SAVAGE-b-ref are on the lower end of this scale, with 0.6/1.3 GB for the 600x HIV mix, 0.9/1.7 GB for the 20 000x HIV mix, and 1.1/3.0 GB for the lab mix, respectively. A complete comparison of runtime and memory usage for all methods is presented in Supplementary Table 4.

2.2.8 Effect of strain divergence and relative abundance

Assembling the sequences of several strains from a viral sample may turn out more difficult depending on both the level of strain divergence and on their relative abundance. After comparing SAVAGE to state-of-the-art methods, we investigated the ranges of divergence levels and of relative abundances that SAVAGE can properly handle, and examined the combined effect of these two parameters on the assembly quality. We used a series of 36 benchmark data sets simulated from two HIV-1 strains: a combination of six divergence levels (from .5 % until 10% of nucleotidic divergence) with six ratios of abundance (from 1:1 until 1:100). We ran SAVAGE-de-novo and SAVAGE-b-ref (i.e., with VICUNA assembled genome). All assemblies were evaluated with MetaQUAST, and Figure 2.3 reports the heatmaps of (A) the coverage fraction of the two genomes, (B) the mismatch rate, and (C) the relative error on the frequency estimates of each strain.

Comparing the two modes of SAVAGE, *de novo* or with a bootstrap reference, we observe similar results and a slight advantage to SAVAGE-de-novo in terms of genome coverage. Altogether, SAVAGE obtains quasispecies assemblies of very low mismatch rates for all divergence levels and all relative abundance ratios, proving its ability to distinguish sequencing errors from true mutations. In general, the target genome coverage is very high for relative abundance ratios starting from 1:1 until 1:10, at all divergence levels. As the relative abundance of the minor strain decreases, it becomes more difficult to reconstruct the corresponding sequence. An extreme relative abundance of 1:100 hinders SAVAGE to reconstruct both strains: genome coverage values around 50% indicate that only one of the two strains has been assembled. We conclude that SAVAGE performs well in both modes (*de novo* and reference-guided) for relative abundances above 1:50 and a wide range of divergence levels.

Capacity to estimate the frequency of each strain The problem of estimating relative frequencies of the contigs assembled for a viral quasispecies is very similar to quantifying the abundances of bacterial genomes from HTS data. Previous work [18] has shown that Kallisto can accurately tackle the latter problem, so we applied this method to our virus contigs as well (see Methods). For the 36 synthetic 'divergence-vs-ratio' benchmarks, we compared the estimated frequency of the minor strain in the sample with the real frequencies. The rightmost panel of Figure 2.3 shows the relative difference between the estimated frequency and the true frequency of the minor strain. This

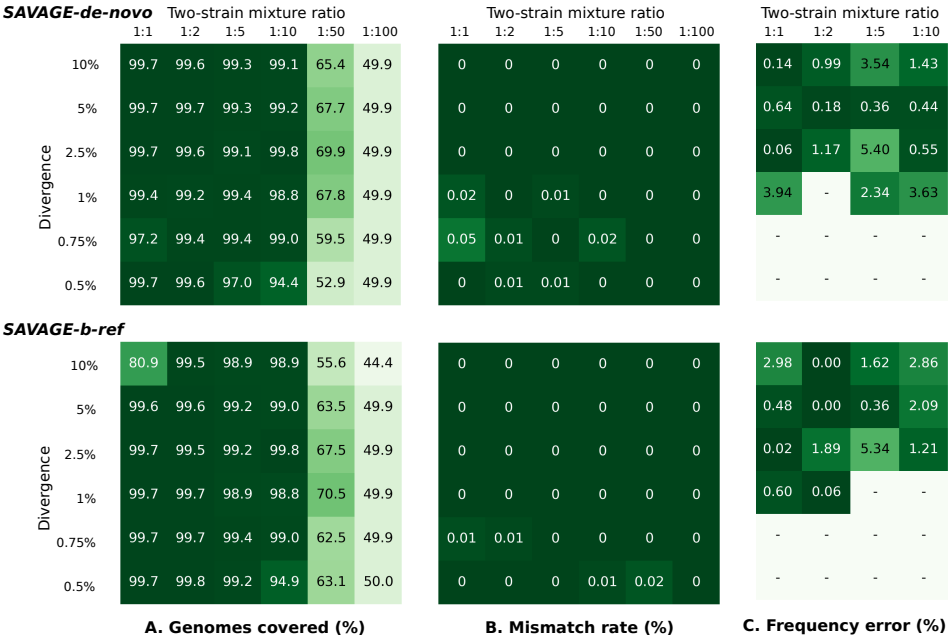

Figure 2.3: Performance of SAVAGE-de-novo and SAVAGE-b-ref, depending on pairwise distance and mixture ratio. **A.** Target genome fraction recovered (%) considering all maximally extended contigs ≥ 500 bp. **B.** Overall mismatch rate (%) considering all maximally extended contigs ≥ 500 bp. **C.** Relative error of estimated frequency for the minor strain (%). Frequency estimates were computed using Kallisto and only assemblies containing exactly two maximally extended contigs longer than 4000 bp were evaluated.

comparison was performed only when the strains were almost fully assembled (exactly two strains of length ≥ 4000 bp), hence abundance ratios of 1:50 and 1:100 were excluded. Of the remaining 24 data sets, 9–10 samples did not satisfy these criteria; the corresponding entries are marked '-' in the heatmaps. Since there are only two strains in the sample, the absolute error is identical for both strains; however, the relative error will be much larger on the low-frequency strain. Hence, we evaluate performance on the most difficult task, namely estimating the frequency of the minor strain. In general, the relative estimation errors are very low: on average 1.65% for the SAVAGE-de-novo contigs and 1.39% for the SAVAGE-b-ref contigs, with an overall minimum of 0% (a perfect estimate) and a maximum of 5.34%.

2.2.9 Zika virus sample

To test SAVAGE-de-novo on real conditions, we ran it on a sample taken from a rhesus macaque infected by an Asian lineage Zika virus [33]. The sequencing reads covered the full ZIKV reference genome used (NCBI sequence KU681081.3) at an average coverage of 30 000x. Using a similar procedure as for the real HIV data (lab mix), we split the reads into patches of approximately 750x each and proceeded with *Stage a* assembly on each patch (Supplementary Fig 1). Subsequently, we used the whole collection of *Stage a* contigs together as input for *Stage b*, which yielded 148 maximally extended contigs longer than 500 bp. A small fraction (4%) of these contigs could not be aligned to the reference genome, but instead matched four human BAC clones (accession AC117500.13, AC002565.1, AC079754.4, and AC015819.5) and one rhesus macaque BAC clone (accession AC190318.8) at > 90% sequence identity, indicating contamination, so we removed them from further consideration. The remaining 142 contigs contained 13 sequences longer than 1000 bp, the largest contig being 1874 bp long, and the N50 measure was 572 bp. The contigs covered the 10767 bp reference genome between positions 225–10767, the greatest divergence occurring between positions 1700 and 4200.

In *Stage c*, we allowed up to 1% divergence between contigs in the overlap graph, thus assembling representatives for groups of very closely related strains (see Methods). This resulted in 6 contigs of length at least 500 bp, now called master contigs. The largest sequence was 4155 bp long and the N50 measure was equal to 2065. Aligning the contigs to the reference genome reveals that the master contigs together form two master strains: their sequences differed only by a one nucleotide deletion at position 4103 followed by a SNP at position 4106 (see Supplementary Fig 2). Our frequency estimation procedure predicted the haplotype harboring the deletion to be the minor haplotype with a frequency of 8.6%, compared to 91.4% for the major haplotype. We hope that in the future, novel external data obtained by different means will become available for this sample, allowing an in-depth validation of our two-strain quasispecies assembly.

2.2.10 Hepatitis C virus sample

Analogous to the ZIKV analysis above, we applied SAVAGE to a hepatitis C patient sample presented in [118]. This sample covers the NS5B region (positions 7602–9374), a gene encoding for the RNA-dependent RNA polymerase, which is essential for viral replication. We found 857 contigs in *Stage b*, with an N50 measure of 533 bp and the largest contig 839 bp long. Aligning the contigs to the HCV reference genome (NCBI sequence NC_004102.1) reveals that the 9646 bp genome was covered between positions 6128–9304, with a relatively constant amount of variation across the whole region. We

observed no contigs resulting from sample contamination (all contigs could be aligned to the reference sequence).

By allowing up to 1% divergence between contigs in the overlap graph in *Stage c*, we continued the assembly. This led to 80 master contigs of length at least 500 bp, of which 5 were longer that 1000 bp. The N50 measure was 535 and the largest sequence counted 1433 bp. Aligning the master contigs to the reference genome shows that one of the master contigs contains a large deletion of 444 bp. This particular sequence could not be aligned across the deletion; instead we found two clipped alignments for the contig, one for the first 781 bases and one for the last 319 bases. Combining these two alignments, the contig covers positions 7723–9267 of the reference genome (nearly the entire NS5B gene), apart from a gap of 444 bases starting at position 8504. The largest master contig spans almost the same region (positions 7923–9356), but it does not show any deletions compared to the reference genome. We conclude that there is a 444 bp deletion in the NS5B gene of only a part of the strains in the sample, in agreement with results from an earlier study [118].

Compared to the previous sample (ZIKV), the current sample shows much more variation in both contigs and master strains. A likely explanation for this is the large difference in numbers of days of infection between the samples: 4 days for ZIKV versus 135 for HCV. To get an estimate on the number of master strains in the HCV sample, we built a conflict graph based on the alignments of the master contigs to the reference genome. An edge in this graph reflects that two contigs disagree on at least one position of the reference genome, hence any clique corresponds to a set of sequences all belonging to different strains. The largest clique in this graph was of size 16, suggesting the existence of at least 16 different strains in the HCV sample.

2.3 Discussion

Recent outbreaks of viral diseases, such as the Ebola or the Zika virus, have pointed out a pressing need for methods to assess the genetic diversity of viral infections in a flexible manner, without strongly depending on the quality of available reference genomes. Here, we have presented SAVAGE, the first method for *de novo assembly of viral quasispecies* based on overlap graphs.

Viral genomes are characterized by high mutation and recombination rates. They are therefore often extreme in terms of both ploidy and the low relative abundance of single haplotypes. In our experiments, existing genome assemblers that do not depend on reference genomes were unable to reconstruct a viral quasispecies completely, where the (often resistance-inducing) low-frequency strains could not be captured sufficiently well. This has pointed out that only more specialized assemblers that can operate without depending on a reference genome have the power to overcome the current

limitations.

We have shown that SAVAGE has this power and thus provides answers to such currently pressing issues. SAVAGE has performed very favorable—if not crucially advantageous—in comparison to a large collection of state-of-the-art de novo assemblers and specialized (but reference-dependent) viral quasispecies assemblers. Thereby, it proved particularly beneficial when being compared to reference-free approaches in terms of reconstructing strains of low frequency, which had been one of the essential goals of this study. Comparisons with existing reference-guided approaches pointed out that those yield contigs that are affected by more sequencing errors in general. Moreover, they tend to become confused by reference genomes of suboptimal quality, while SAVAGE behaves in a robust manner and can also make favorable use of such suboptimal bootstrap (ad-hoc) reference genomes. Last but not least, our method significantly outperforms the only available *de novo* viral quasispecies assembler (MLE-Haplo) in terms of assembly quality, runtime, and memory usage. In an overall account, SAVAGE has proven to bridge a significant gap in the spectrum of viral quasispecies assembly approaches.

We believe that the central methodical reason for the benefits of our approach is the use of overlap graphs as the underlying assembly paradigm. While assembling genomes of low ploidy usually works favorably based on de Bruijn graphs, we have pointed out that using reads at their full length is key in assembling viral quasispecies, where distinguishing between low-frequency mutations and sequencing errors is imperative. The key insight is that (genetically linked) true mutations co-occur among different reads. Examining the full read span decisively enhances the detection of patterns of co-occurrence. Beyond enabling the detection of low-frequency strains, this also allows correction of sequencing errors in novel ways. We have pointed this out by making integrative use of sound statistical sequence models in combination with an iterative algorithmic scheme, which extends reads into contigs of increasing length and extremely low error content.

Key to reference free construction of overlap graphs has been the use of FM-index based techniques, which has been novel in the context of the analysis of viral data. Moreover, we have demonstrated that overlap graphs also seem to be the approach of choice when aiming to make use of ad-hoc consensus reference genomes, such as provided by specialized tools that construct a single consensus sequence from patient sample read data. Often, the resulting consensus sequence is of worse quality than a well-curated reference sequence. This can substantially disturb approaches that rely on the underlying reference as a sequence template (e.g. PredictHaplo, ShoRAH). Overlap graphs constructed by making use of reference sequence coordinates provide a robust alternative, since they use the reference sequence only as a coordinate system for the determination of overlaps.

A few more things are noteworthy. First, the bootstrap reference approach SAVAGE-b-ref has proven to outperform reference guided approaches in terms of the error rates of the contigs, even when they make use of high-quality reference sequence, which further underlines the general use of overlap graphs. Second, the target genome coverage of our full *de novo* approach SAVAGE-de-novo exceeded that of the high-quality reference-guided approaches, which points out its ability to distinguish sequencing errors from true mutations. Finally, SAVAGE-b-ref also depends on the quality of the reference sequence: the target genome coverage is 13.6 points lower compared to SAVAGE-h-ref on the 10-strain HCV mix. This, of course, had to be expected: if reference coordinates are too mistaken, overlaps cannot be detected. This last point underscores that a full de novo approach can come with decisive extra advantages.

Of course, there is still room for improvements. While substantially faster and more space efficient than previous overlap graph based viral quasispecies assembly algorithms, SAVAGE has been particularly tailored towards dividing deep coverage data sets into chunks of 500 to 1000x, and merging the contigs of the chunks in subsequent steps, because this reflects its statistical calibration. While this works well, it sets certain limits on the frequency of strains it can recover — haplotypes of frequencies below 1% remain difficult to reconstruct. In future work, we will seek to lower these limits further by considering novel strategies for computing cliques in overlap graphs. On the algorithmic side, we will also explore alternative indexing techniques that allow for more relaxed definitions of overlaps and faster computation. Last but not least, incorporating long read data into SAVAGE may help to reconstruct full-length genomes.

2.4 Methods

2.4.1 Overlap graph construction

We first provide a brief definition of an overlap graph and then sketch how to construct such graphs from patient sample read data using *indexes* or *reference genomes* as two options.

Overlap graphs. For a collection $\mathscr{R}$ of sequencing reads (*Stage a*) or contigs (*Stages b,c*), both of which are sequences over the alphabet of nucleotides $\{A, C, G, T, N\}$ (which includes N as a common placeholder for unknown nucleotides), the *overlap graph* $G = (V, E)$ is a directed graph, where vertices $v \in V$ correspond to reads/contigs $R \in \mathscr{R}$ and directed edges connect reads/contigs $R_j, R_j \in \mathscr{R}$ whenever a suffix of R_i of sufficient length matches a prefix of R_j and $\mathrm{QS}(R_i, R_j) \geq \delta$ where $\mathrm{QS} : V \times V \to \mathbb{R}$ is a quality score that has to exceed a certain threshold δ. For *Stages a, b* we make use of the statistical model presented in [118], where $\mathrm{QS}(R_i, R_j) \geq \delta$ reflects that the overlapping parts of reads R_i and R_j present a locally identical haplotypic sequence. Note that the statistical

model includes a refined analysis of the (Phred-scaled) error profiles that underlie R_i and R_j so as to reflect that sequencing is an erroneous process and hence to assess the identity of their overlapping parts on a sound statistical basis.

In *Stage c*, $QS(R_i, R_j)$ reflects the fact that the two contigs share only a limited amount of mismatches in their overlaps, meaning that they did likely emerge from identical master strain sequences.

Paired-end reads. SAVAGE was designed for short reads (typically Illumina reads); after merging self-overlapping pairs, the input in *Stage a* may contain paired-end reads and/or single-end reads. To make use of the pairing information, we add another edge restriction by allowing only the overlap cases shown in Figure 2.4. For overlaps involving a paired-end read, we require both read ends to have a sufficiently long overlap (at least half of the minimum overlap length for single-end reads) as well as a sufficiently high quality score.

Construction. Construction of overlap graphs always proceeds in two steps. First, pairs of reads (R_i, R_j) are determined that have a sufficiently long and well-matching overlap. Subsequently, QS is evaluated on all pairs (R_i, R_j). For *Stages b and c*, where the input is sufficiently small, the first step is implemented by pairwise comparison of all contigs using BLAST [5]. The only difficulty is the first step in *Stage a*, where the input is very large (the original deep coverage data). This requires some sophistication; we explore two options:

1. With a read index: We determine all sufficiently long overlaps between sequencing reads using FM-index based techniques [121, SFO] such that overlaps contain at most 2% mismatches (accounting for up to 1% sequencing errors in each of the reads). This method, however, only works on single-end reads, so we first ignore the paired-end relations and consider each of the sequences as a single-end read. Then, after listing all pairwise overlaps with SFO, we reconsider the pairing information, outputting only overlaps that are supported by both read ends as described above.

2. With a reference genome: We align all reads against a reference genome; here we may use an ad-hoc consensus genome obtained by running an assembly tool on the sample reads. With all read alignments in hands, it is then computationally straightforward to determine all sufficiently long and sufficiently matching overlapping pairs.

Read orientations. When merging multiple reads into one consensus sequence, it is important that the reads agree on their respective orientations. Therefore, we apply a read orientation routine that assigns a label $(+/-)$ to every read, indicating the orientation in which its sequence should be considered. This routine starts by setting the orientation of a node of minimal in-degree to $+$, then recursively labels all out-neighbors as defined by the corresponding edges (Figure 2.5, panel A). When there is no perfect labelling possible, meaning that there are conflicts among the read orientations due to inversions, we heuristically search for an orientation that leads to a

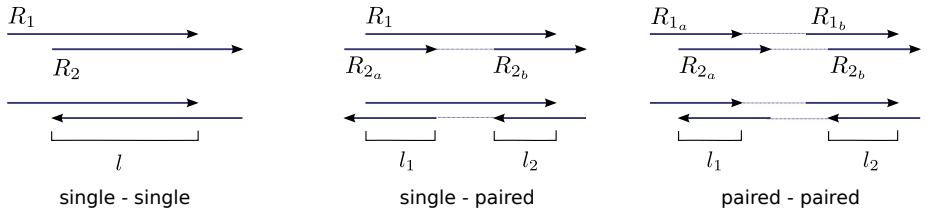

Figure 2.4: Edge criteria. For an overlap to become an edge in the overlap graph, it must satisfy three criteria. First, the overlap length l must be at least the minimal overlap length L. Second, the overlap quality score $QS(R_1, R_2)$ must be at least the minimal score δ. For overlaps involving paired-end reads, we require both $l_1 \geq L$ and $l_2 \geq L$, and, analogously, $QS(R_{1_a}, R_{2_a}) \geq \delta$ and $QS(R_{1_b}, R_{2_b}) \geq \delta$. Finally, we only accept overlaps where the sequence orientations of a paired-end read agree: either both sequences in forward orientation, or both sequences in reverse orientation.

minimal amount of conflicts among the reads.

2.4.2 Overlap graph based assembly

In all stages, our algorithm proceeds as an iterative procedure where contigs grow with the iterations. The final contigs (in particular the output of *Stage b*, or, optionally, *Stage c*) can substantially exceed the length of the original reads. As our analyses demonstrate, these contigs present haplotype specific sequences with high accuracy.

Cliques and contigs. The main idea of our algorithm is to compute cliques in the overlap graph. A *clique* is a subset of the nodes such that each pair of nodes is linked by an edge. By definition of the edges, a clique groups reads that stem from identical haplotypes. Within a clique, reads/contigs share (possibly low-frequency) true mutations while sequencing errors are not shared by the majority of reads (Figure 2.1, Panel B). Hence, cliques can be used to clearly distinguish between true mutations and sequencing errors. This further allows us to correct these errors by transforming cliques into contigs that represent an error-corrected consensus sequence of the reads in the clique.

Transitive edge removal. The number of maximal cliques in an overlap graph grows exponentially with the number of nodes in the graph, that is here, with the read coverage of the data set giving rise to the overlap graph. While our method relies on cliques for the purpose of error correction, the size of the cliques does not have to exceed a certain threshold for that goal.

A common approach to reduce the complexity of an overlap graph is to remove transitive edges—see e.g. [110]. An edge $u \rightarrow w$ is called *transitive* if there exist a vertex v

and edges $u \to v$, $v \to w$. We call an edge $u \to w$ *double transitive* if there exists a vertex v and *transitive* edges $u \to v$, $v \to w$, illustrated in Figure 2.5, panel B. Note that, by definition, any double transitive edge is also single transitive. We found that removing double transitive edges bounds the size of the cliques to 4, thus decisively limiting the number of maximal cliques and allowing efficient maximal clique enumeration, while still allowing for safely distinguishing errors from true mutations.

To find all double transitive edges, we first remove all non-transitive edges from the overlap graph to obtain the transitive graph G'. This can be done efficiently by computing the inner product of a_u^- and a_v^+ for all pairs $(u, v) \in V \times V$, where a_u^- (resp. a_v^+) is the adjacency vector of outgoing (resp. incoming) edges of u (resp. v). Applying this procedure to G we obtain G', and to find all double transitive edges we apply the same procedure to G'.

In the first iteration of *Stage a*, we remove all double transitive edges from the overlap graph. This reduces the number of contigs obtained in this iteration by an order of magnitude, leading to a decrease in CPU time and memory usage of even two orders of magnitude (Supplementary Table 4). In later iterations our algorithm no longer depends on clique formation because the reads (contigs) are assumed to be already of high quality. This allows us to remove not only double but also single transitive edges.

Read clustering. In the first iteration of *Stage a*, we cluster reads by enumerating maximal cliques in the overlap graph. After double transitive edge removal in an acyclic graph, the maximum clique size is 4, as illustrated in Figure 2.5, Panel B: a clique of size 5 will always use a double transitive edge. In practice, our overlap graphs are nearly acyclic and all cliques are of size at most 4. This implies that the total number of cliques is polynomial in the number of nodes, hence we can efficiently enumerate all maximal cliques; we use the degeneracy algorithm presented in [38] to do so. For the error correction algorithm to function optimally, we solely consider cliques of size 4 in this iteration.

In later iterations, after removing all single transitive edges, we merge pairs of contigs into new (extended) contigs. This does not require clique enumeration of any kind. See Figure 2.5, panel C for an illustration of the two read clustering techniques. In case of conserved regions among multiple strains, there can be branches in the overlap graph. In such situations it is often impossible to connect the variants left and right of the conserved region, hence we do not merge any pair of contigs connected by a branching edge (Supplementary Fig 3).

Contig formation and error correction. We transform all reads/contigs within a cluster (a clique or a pair of contigs) into a consensus sequence. It is important to determine the consensus very carefully, because the original sequencing reads may contain up to 1% sequencing errors. Every consensus base is determined by a position-wise weighted majority vote, where the weights correspond to the respective base quality

scores, as described in [118] and Supplementary Methods. This procedure was designed to correct for all putative sequencing errors showing among members of a clique, which is especially relevant in the first iteration of *stage a* (the error correction step). In this specific iteration, therefore, we require cliques of size at least 4; it is then highly unlikely that all the reads in a clique will agree on a sequencing error. We remove the extremities of the resulting contig where the support of the clique is less than 4 (Figure 2.5, panel D). Reads that are not contained in any size 4 clique are discarded after this iteration.

Graph updating. The newly constructed contigs become the nodes of the updated overlap graph and we need to determine the edges between those nodes. In other words, we need to find all pairs of contigs satisfying our overlap criteria. In *Stage a*, we examine all pairs of contigs that share an original read. This approach is very efficient, but risks ignoring overlaps of contigs that do not share an original read. In *Stages b* and *c* the graph is sparse enough, such that we can update the edges by considering all induced overlaps. This means that for every edge $u \to v$ in the graph before updating, we consider every overlap $u' \to v'$ for all $u' \in S_u, v' \in S_v$, where S_u, S_v are the sets of all newly constructed contigs containing u, v, respectively. In addition, we also reconsider all overlaps that were not included as an edge in the graph before updating due to an insufficient overlap quality score.

Iteration. The key idea of the SAVAGE assembly algorithm now is to repeatedly apply this twofold procedure of clique enumeration (*Stage a*) or merging pairs (*Stages b and c*) and contig formation. Thereby, all contigs of iteration $i \geq 1$ become nodes in the overlap graph of iteration $i+1$, which results in an overlap graph to be processed in iteration $i+1$. We repeat this procedure until there are no more edges in the overlap graph. Key to success is that contigs are constantly growing along the iterations, and, upon convergence, greatly exceed the length of the original reads. An example of the progression of contig lengths during the three stages of the algorithm is given in Supplementary Table 5.

2.4.3 Parameter settings

There are three parameters to be set, namely, the overlap score threshold δ, the mismatch rate mr allowed in the overlaps, and the minimal overlap length L. To analyze the behaviour of the overlap score function, we simulated 2×250 bp Illumina MiSeq reads from different genomes, diverging between 1% and 10%. We computed all overlaps among those reads and classified them by the number of true mutations in the overlap (not counting mismatches that are due to sequencing errors). This resulted in distributions P_i, $i \geq 0$, representing the overlap scores found in case of i true mutations (Supplementary Fig 4), from which we concluded that $\delta = 0.97$ is the optimal choice. To be more conservative, this threshold can be raised, but this comes at the cost of a

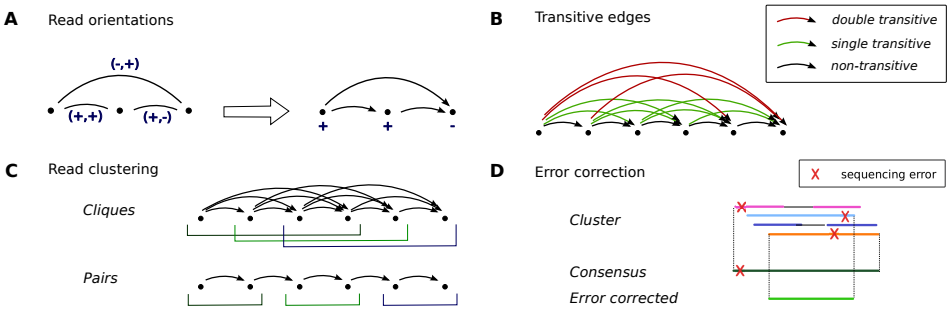

Figure 2.5: Algorithmic details. **A.** Read orientations: Given an edge $u \to v$ with orientations $(-,+)$. Then if u is labelled $+$, the induced label for v is $-$, while if u is labelled $-$ the induced label for v is $+$. This procedure leads to a vertex labelling in $O(V)$ time. **B.** Transitive edges: An edge $u \to w$ is called *single transitive* (resp. *double transitive*), shown in green (resp. red), if there exists a vertex v and edges (resp. *transitive* edges) $u \to v$, $v \to w$. **C.** Read clustering by cliques (top) or by pairs (bottom). **D.** Error correction: when a consensus sequence is constructed from a cluster of reads, the extremities are removed.

decrease in the target genome coverage.

The mismatch rate parameter allows overlaps having an insufficient overlap score to become edges in the overlap graph if the mismatch rate is sufficiently low. By default, this parameter is set to 0, meaning that we only rely on the overlap score for constructing the overlap graph. When assembling master strains, however, the allowed mismatch rate was set to 0.01, so that strains diverging by less than 1% were merged into a consensus sequence.

Finally, the setting of the minimal overlap length parameter depends on the average read coverage and sequencing depth. Increasing the minimal overlap length results in a faster algorithm and lower error rate, because the overlap graph will be very much restricted. But this achievement comes with a potential loss of low-frequency strains, since the corresponding reads may not have sufficiently long overlaps. In general, we found a minimal overlap length of 50–70% of the total read length to work well. The exact command lines and parameter settings used for all experiments can be found in Supplementary Methods.

2.4.4 Frequency estimation

We apply Kallisto [18] to estimate relative frequencies of the contigs assembled for a viral quasispecies. Kallisto was designed for quantifying the abundances of bacterial genomes from HTS data, which is similar in spirit to estimating frequencies for viral quasispecies assembly. The Kallisto algorithm takes as input the original sequencing reads along with the contigs, and returns for every contig a so called TPM (Transcripts Per Million). This number estimates the amount of sequencing reads corresponding to this contig for every one million reads considered, and it is independent of the contig length. We translate these counts to relative frequencies by dividing each TPM by the sum of TPMs of all contigs evaluated. For the heatmaps in Figure 2.3, panel C, we only evaluated the two contigs of at least 4000 bp.

2.4.5 Other methods used for evaluation

For benchmarking, we compared SAVAGE against the state-of-the-art approaches ShoRAH [130] and PredictHaplo [97]. Both methods were run with default parameter settings, after aligning the reads to the reference genome using BWA-MEM [63]. The *de novo* assembler MLEHaplo [72] required the reads to be error corrected first, for which we used MultiRes [71] with default settings (recommended by the authors). Unfortunately, we could not compare against VGA [73] and HaploClique [118] because these software packages were no longer maintained.

2.4.6 Data simulations

To evaluate performance of SAVAGE, we designed several simulated data sets. We used the software SimSeq [112] to simulate Illumina MiSeq reads from the genome of interest. In order to obtain reads similar to the real 5-virus-mix data, we simulated 2×250 bp paired-end reads, with a fragment size of 450 bp and the MiSeq error profile provided with the software. In addition, we also simulated a 5-strain HIV mix with shorter reads (2x150 bp). The genomes used for each data set are listed in the Supplementary Methods.

2.4.7 Read trimming and merging

Before running any of the methods, the raw Illumina reads were trimmed using CutAdapt [76]. Next, we applied PEAR [132] for merging self-overlapping read pairs. This resulted in a final read set containing both single-end and paired-end reads, on which we ran SAVAGE. For the other methods (MLEHaplo, PredictHaplo, ShoRAH, and VICUNA) we used the trimmed reads without merging, since neither of these methods

accepts a combination of single- and paired-end reads. In addition, MLEHaplo required an error correction step on the input reads which was performed using MultiRes [71].

2.4.8 MetaQUAST evaluation

We use MetaQUAST [82] for quality evaluation of the assembled contigs, which evaluates the contigs against each of the true viral genomes. By default, MetaQUAST uses the option `-ambiguity-usage all`, which means that all possible alignments of a contig are taken into account. However, the genomes in a viral quasispecies can be so similar that a contig may align to multiple strains, even though it only matches one haplotype. Therefore, we manually changed this option to `-ambiguity-usage one`, such that for every contig only the best alignment is used. Contigs shorter than 500 bp were ignored during evaluation.

2.4.9 Software availability and data access

A C++ implementation of SAVAGE is available for public use as part of the HaploConduct package at https://github.com/HaploConduct/HaploConduct. All simulated data sets can be downloaded from https://bitbucket.org/jbaaijens/savage-benchmarks.

CHAPTER 3

OVERLAP GRAPH-BASED GENERATION OF HAPLOTIGS FOR DIPLOIDS AND POLYPLOIDS

In this chapter we present POLYTE (POLYploid genome fitTEr) as a new approach to de novo generation of haplotigs for diploid and polyploid genomes of known ploidy. We adapt the overlap graph-based assembly approach described in the Chapter 2 to be applied in a scenario of fixed ploidy and low to medium sequencing depths (10–100x). In order to deal with low coverage sequencing data, edge constraints for the overlap graph are less restrictive in comparison to the previous chapter. As this increases the number of spurious edges and thereby the risk of assembling false haplotypes, we present a procedure to reduce the number of spurious edges in the overlap graph.

Experiments on both real and simulated data demonstrate that POLYTE establishes new standards in terms of error-free reconstruction of haplotype-specific sequences. As a consequence, POLYTE outperforms state-of-the-art approaches in various aspects, where advantages become particularly distinct in polyploid settings.

Published as:
J.A. Baaijens and A. Schönhuth. Overlap graph-based generation of haplotigs for diploids and polyploids. *Bioinformatics*, btz255, 2019 (in press).

Supplementary material: https://doi.org/10.1093/bioinformatics/btz255.

3.1 Introduction

In most eukaryotic organisms genomes come in copies, where each copy stems from one of the ancestors. The number of copies determines the *ploidy* of the organism: while *diploid* relates to two copies, *polyploid* refers to more than two copies.The copy-specific sequences are referred to as *haplotypes*, which generally differ in terms of the genetic variants affecting them. Distinguishing the two haplotypes in diploid organisms (such as in most vertebrates) or more than two in polyploid organisms (such as many plants and some funghi) plays an important role in various disciplines. Prominent examples are genetics, where assigning variants to ancestors is key [114], and medicine, because very often haplotype-specific combinations of variants establish clinically relevant effects, e.g. when disease risks have been inherited [45]. In general, determining *haplotypic sequence*, i.e. keeping track of ancestry-based dependencies is instrumental in many biomedical settings.

Assembling the two (diploid) or more (polyploid) haplotypes from sequencing reads is known as *haplotype-aware genome assembly*, and the resulting assembled pieces of sequence are *haplotigs*, as a shorthand for haplotype-aware contigs. The advent of next-generation sequencing (NGS) has brought about a plethora of NGS read compatible assembly programs. The vast majority of these programs, however, do not yield haplotigs, but consensus genome sequence, as a summary across all haplotypes involved. Even then, sequencing errors, read length and hardware limitations already pose fundamental challenges during the assembly process.

Generating haplotigs from NGS reads—which is the challenge that we tackle here—comes with additional obstacles. Beyond distinguishing between errors and true sequential variants, one needs to assign the true sequential variants to the different copies. This requires keeping track of information that allows to link the true sequential variants stemming from identical copies. However, NGS reads in general are rather short: techniques are needed that can link haplotype-specific variants across read boundaries. Despite the many recent advances, this is not (yet) a standard procedure in genome assembly: haplotype-aware assembly can still be considered in its early stages of development which explains that further advances are desirable.

Motivation. The majority of sequencing machines installed worldwide perform traditional NGS, such as Illumina sequencing. A plethora of population-scale sequencing studies (e.g. [14, 113, 115, 116]) have filled up databases with traditional, short NGS reads. In terms of quantities, traditional short NGS reads exceed the amount of reads stemming from more recent third-generation sequencing (TGS) protocols by at least one order of magnitude. The increase in read length due to TGS has considerably spurred the development of methods for haplotype-aware assembly (see Related work).

While the increase in length is beneficial, the increase in sequencing error rates is also a major obstacle when distinguishing between haplotypes, usually leaving applicants with ambiguities that are hard to resolve.

Recent work has pointed out that targeted examination of NGS (Illumina type) reads can have significant positive effects in haplotype-aware assembly [12, 91]. Seemingly, the enormous quantities of traditional NGS read data have been underexploited in terms of haplotig computation so far. This establishes our major motivation.

To better understand where serious progress can be made, one needs to realize that existing methods for haplotype computation from traditional NGS (Illumina) reads fall into two classes: the first (and arguably more popular) choice of approaches are referred to as *haplotype assembly* programs. These approaches make use of a reference genome to call variants from aligned reads, which are subsequently phased into separate haplotypes. The advantage of haplotype assembly programs is their stability and their resource-friendly usage. Examples for *diploid haplotype assembly* are WhatsHap [91], Phaser [20], HapCut2 [36], ProbHap [60] and HapCol [95]. Examples for *polyploid haplotype assembly* are HapCompass [2], HapTree [12], SDhaP [29], and H-PoP [128]. The disadvantage of haplotype assembly programs is that they depend on high-quality reference sequence as a backbone. In addition, they depend on external variant call sets. These two factors can introduce non-negligible biases.

The second class of methods is *de novo haplotype-aware assembly* approaches that can deal with traditional NGS (in particular Illumina) reads. The advantage of such approaches is that they are independent of reference genomes and external call sets, which eliminates the externally induced biases. There are only little such approaches available however; to the best of our knowledge, only ALLPATHS-LG [100], Platanus [55], and dipSPAdes [104] explicitly aim at computation of haplotigs from (diploid) NGS data. However, ALLPATHS-LG and Platanus require particularly tailored libraries, which renders their general application difficult, and the dipSPAdes software is no longer maintained. In results of ours, we further noted that SPAdes [9] can be run in diploid mode (which is not to be confused with the no longer maintained dipSPAdes), and is able to compute haplotigs (surprisingly not only in diploid, but also in conventional mode), thereby likely establishing the only tool among the (myriad of) approaches for consensus oriented genome assembly (see [17, 105] for references) that one can use for computation of haplotigs from short NGS reads.

In summary, there are no approaches that 1) specialize in the generation of (high-quality) haplotigs, but 2) do not depend on high quality reference sequence as a backbone, 3) do not depend on external variant call sets and 4) do not require particularly tailored sequencing libraries.

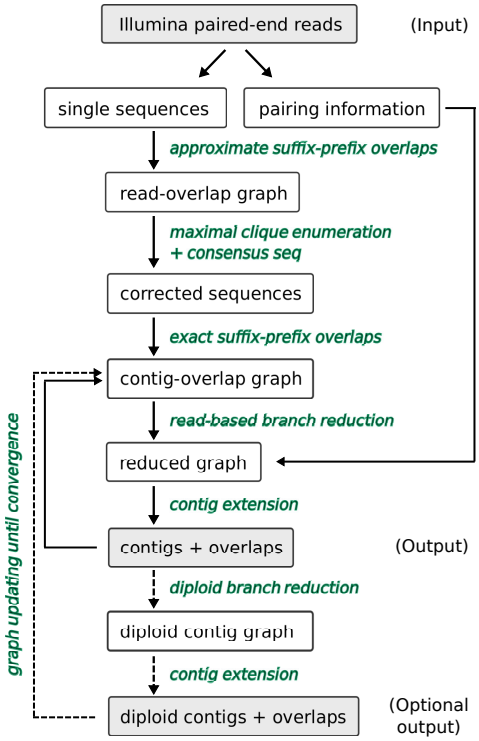

Figure 3.1: Algorithm overview.

Contribution. The contribution of this paper is to close this gap in the landscape of approaches. We present POLYTE (POLYploid genome fitTEr), as an approach to do this for genomes of known ploidy. Our results indicate that POLYTE outperforms state-of-the-art approaches of the two classes—haplotype assembly and de novo assembly approaches—with significant advantages in a variety of relevant aspects. As an example of an application scenario, POLYTE outperforms the other approaches in reconstructing individual haplotypes of the human Major Histocompatibility Complex (MHC). This region of 6Mb on chromosome 6 is essential to the acquired immune system and shows very high genetic variability; haplotype-aware reconstruction of the MHC region therefore usually is particularly challenging during the assembly process. Note finally that the majority of approaches focuses on diploid genomes. Therefore, the lack of approaches that can compute haplotigs for organisms of ploidy larger than two is even more striking. For ploidy larger than two, POLYTE achieves performance rates that are nearly on a par with those achieved for diploid organisms. To the best of our understanding, because of the lack of competitors, one might perceive POLYTE's achievements for polyploid organisms as a novelty in its own right.

Related work. In terms of assembly paradigms, POLYTE is an *overlap graph-based* approach. It adopts ideas from earlier work that either focused on variant discovery [75], viral quasispecies assembly [7, 118] or metagenome gene assembly [46] and unites the virtues of Marschall et al. [75]—the ability to handle low coverage—on the one hand, and Baaijens et al. [7] and Töpfer et al. [118]—dealing with real overlap graphs and contig computation—on the other hand. That is, POLYTE brings forth an iterative overlap graph-based scheme for contig generation that reliably works in *low coverage settings*, requiring coverage of only as low as 5x per haplotype.

Note finally that our approach also draws motivation from the recent technology shifts, such as the advent of third-generation sequencing (TGS) and explicitly haplotype-aware sequencing protocols like StrandSeq [96], which have put the computation of haplotigs into the focus of current attention. Chin et al. [23], Jain et al. [53] and Weisenfeld et al. [127] describe approaches that aim to exploit the respective advances in sequencing technology and protocol design. Although there are similarities between these approaches and POLYTE, we focus on NGS data and hence our method fully exploits paired-end read information. We consider the adaptation of POLYTE to TGS data most interesting future work: the framework of POLYTE is generic in terms of choosing reads, such that this is a matter of adapting parameters, more than anything else. We recall, however, that our motivation was to bring forward a method that exploits (the abundantly available) traditional NGS reads in the first place. This, e.g. enables to reconstruct MHC region haplotypes in various population-scale studies (e.g. Besenbacher et al. [14], Sudmant et al. [113], The Genome of the Netherlands Consortium [115] and The UK10K Consortium [116]), which has been a major challenge so far.

3.2 Methods

We present POLYTE, an algorithm to assemble individual haplotypes of diploid and polyploid genomes from short read sequencing data; see Figure 3.1 for the complete workflow. POLYTE follows the overlap-layout-consensus (OLC) paradigm, where consensus refers to removing errors within haplotypes (instead of the common interpretation of reaching consensus across different haplotypes). Our method starts by constructing a *read-overlap graph* which is used for error correction of the input sequences. Subsequently, we make use of an iterative OLC scheme, where in each iteration a *contig-overlap graph* is constructed. This graph is further reduced by applying *transitive edge removal* and *read-based branch reduction*. Then, contigs are clustered and merged according to their interplay within the overlap graph, resulting in a collection of extended contigs ('contig extension' in Figure 3.1). These extended contigs establish the nodes of the contig-overlap graph of the next iteration, which is achieved by an updating procedure. When contigs can not be merged any further, POLYTE outputs the final set

of contigs. When dealing with diploid organisms, an additional assembly stage can be activated which consists of two additional steps ('diploid branch reduction' and 'contig extension' in Figure 3.1), creating an optional output that is refined for diploid organisms.

Given that we are dealing with data of relatively low sequencing depth, we need to exploit the information present in the sequencing reads as much as possible. The initial error correction procedure is particularly crucial, as sequencing errors can heavily disturb the process of distinguishing between different haplotypes. For this error correction step, approximate suffix-prefix overlaps are computed to establish an initial read-overlap graph. Inspired by Baaijens et al. [7] and Töpfer et al. [118], maximal cliques are enumerated in the non-oriented graph and errors are corrected by inspecting the overlaps between reads within the cliques. By design of the overlap graph—edges indicate that two reads stem from identical haplotypes—every clique only contains reads from identical haplotypes, which allows to eliminate errors based on majority votes. Note that this procedure is particularly tailored to low coverage settings with known ploidy: admissible clique sizes and minimal sequence overlap lengths can heavily vary in comparison to earlier approaches. However, with edge criteria that are much less restrictive than in other approaches, we obtain a larger number of spurious edges. We have developed a procedure for *read-based branch reduction* to reduce the number of spurious edges in the overlap graph, which is of great importance for accurate reconstruction of haplotigs.

In the following sections we will discuss each of the steps involved in POLYTE, following the workflow depicted in Figure 3.1.

3.2.1 Read-overlap graph construction

The steps outlined in this section refer to the initial step 'approximate suffix-prefix overlaps' that leads to the establishment of the 'read-overlap graph' in Figure 3.1.

Read-overlap graph: definition. The read-overlap graph follows the idea that nodes are reads and edges indicate that a pair of reads stem from identical haplotypes. Given the input consisting of paired-end sequencing reads (Illumina), let $\mathscr{R}$ be the collection of single-end sequences from all paired-end reads. The *read-overlap graph* $G = (V, E)$ is a directed graph where V corresponds to the collection of input sequences $\mathscr{R}$. That is, for every paired-end read we have two vertices $v, v' \in V$, one for each single-end sequence $R \in \mathscr{R}$. Directed edges $v_i \to v_j \in E$ connect sequences R_i, R_j whenever the suffix of R_i overlaps the prefix of R_j for at least 50% of the average sequence length of all reads. Furthermore, for each edge $v_i \to v_j$, we require $QS(R_i, R_j) \geq \delta$, where $QS : \mathscr{R} \times \mathscr{R} \to \mathbb{R}$ is a quality score and δ is an appropriate threshold. This threshold is determined based on empirical statistics so as to maximize the chances that the

edge (v_i, v_j) indeed indicates that the corresponding sequences R_i and R_j stem from identical haplotypes; increasing the δ-threshold would lead to a higher accuracy but possible loss of low abundance haplotypes. In this, we largely follow ideas presented in earlier work [7, 75, 118].

The difference with respect to these prior approaches is that only single ends are considered, whereas in the earlier approaches nodes represent the entire paired-end reads. Also note that here overlap graphs are twice as large in comparison to the earlier approaches, because each paired-end read is represented by two nodes, instead of only one. While this difference imposes substantial methodical and technical challenges, it is key to dealing with low coverage because it decisively increases the recall in terms of recovering reads that stem from identical haplotypes. However, it also implies follow-up complications, because the information that read ends come in pairs is temporarily lost. In POLYTE, paired-end information is stored and used in later steps; see Section 3.2.4 below.

Construction. Computation of the edges for the read-overlap graph requires enumeration of all pairwise approximate suffix-prefix overlaps (of sufficient length) between the single read ends $R \in \mathscr{R}$ and evaluation of a quality score $QS(R_i, R_j)$ for each pair of sequences for which a sufficiently good overlap was established during the approximate suffix-prefix overlap computation. We further orient the edges (which is necessary because reads can stem from either the forward or the reverse stand) and systematically remove double transitive edges, which ensures that one can enumerate maximal cliques in an efficient manner (see Section 3.2.2). Each of these graph construction steps is described in detail in Section 1 of the Supplementary Material.

The computation of approximate suffix-prefix overlaps for vertebrate genome-sized input read sets is a serious issue, currently hardly conceivable without external auxiliary means (see also Simpson and Durbin [110]). Here, we suggest a method that aims to suppress externally introduced biases to a maximum degree. We make use of a reference genome for binning reads in an initial step and, after binning, we discard the reference genome and any related information entirely such that POLYTE operates in full de novo mode. This binning step does not require a high-quality reference genome, as long as reads get mapped; any unaligned reads are discarded (see also Supplementary Section 9).

3.2.2 Correction of sequencing errors

After the establishment of the read-overlap graph, we cluster its nodes by enumerating the *maximal cliques* contained in the non-oriented graph. The idea is to collect groups of reads belonging to the same haplotype and produce error-free sequences for subsequent assembly steps ('corrected sequences', Figure 3.1). By definition of

a maximal clique—a maximal group of nodes all of which are connected by edges—maximal cliques represent maximum-sized groups of reads all of which belong to the same haplotype. Once all maximal cliques are determined, it is therefore reasonable to merge the reads within a maximal clique into a single contig. Note that this contig is longer than the individual reads participating in the contig and that sequencing errors can be eliminated by raising majority votes among the reads participating in the maximal clique. While this reflects an approved procedure in its generic form [7, 46, 118], accounting for the particular setting we are facing here—namely low coverage in combination with sequence-based edge definition—requires particular care.

The minimum clique size depends on the coverage per haplotype; in all settings considered we are dealing with known ploidy, while the overall coverage of reads can be determined by usual considerations, which yields per-haplotype-coverage estimates. To determine the optimal minimum size of a clique for a given per-haplotype coverage, we compute the probability $p_{c,k}$ that, due to unfortunate fragmentation of sequencing reads, at a per-haplotype coverage of c there is no clique of size k that extends a given sequencing read R to the right when requiring at least 50% read overlap. In other words, we compute the probability that there are *at most* $k-1$ reads extending R to the right; the exact same analysis applies to extensions to the left.

For determining $p_{c,k}$, we assume that sequencing reads are fragmented randomly, which implies that reads are generated independently of one another. Let R be a read and S be a set containing reads from the haplotype of R at exactly 1x coverage, further assuming that all reads $R' \in S$ have the same length as R (which reflects that all single read ends have the same length). It is straightforward to see that the probability that there is $R' \in S$ that overlaps R for at least 50% of its length (into one direction, left or right) as 0.5. When dealing with a per-haplotype coverage of c, we assume the existence of c sets of reads $S_i, i = 1,..,c$ all of which contain reads that cover the haplotype c at 1x. For computing $p_{c,k}$ we consider that for only $k-2$ of the c sets $S_i, i = 1,...,c$ we have that there is $R' \in S_i$ that overlaps R for at least 50% of its length (resulting in a clique of size at most $k-1$), which evaluates as

$$p_{c,k} = \sum_{i=0}^{k-2} \binom{c-1}{i} 0.5^i \, 0.5^{c-1-i} = \sum_{i=0}^{k-2} \binom{c-1}{i} 0.5^{c-1}. \tag{3.1}$$

We aim to have $p_{c,k}$ low to be able to deal with sufficiently many cliques, hence for every choice of c we compute k such that $p_{c,k} < 0.001$. In this regard, we obtain that for up to 10x per haplotype an appropriate choice for the minimum clique size is 2, for coverages between 10x and 15x a minimum clique size of 3 is required, while for $c \geq 15$x an optimal choice for the minimum clique size is 4. Note that in practice cliques do not grow larger than size 4 because of double transitive edge removal (see Supplementary Material).

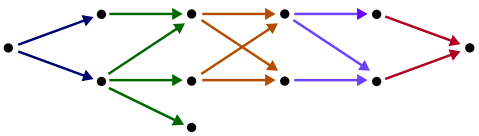

Figure 3.2: Illustration of branching components in a contig-overlap graph. Edges of the same colour belong to the same branching component.

3.2.3 Contig-overlap graph construction

Given the corrected sequences obtained by merging maximal cliques, we build a new graph: the contig-overlap graph (see Figure 3.1).

Contig-overlap graph: definition. The contig-overlap graph $G' = (V', E')$ is very similar to the read-overlap graph, except that we construct it from a set of contigs assumed to be free of sequencing errors. Therefore, every node $v \in V'$ corresponds to a contig and we add an edge between a pair of nodes whenever they have an exact (i.e. error-free) overlap of sufficient length.

Construction. The contig-overlap graph can be constructed very efficiently by making use of the FM-index-based algorithm from Chapter 2 while allowing only exact overlaps. This gives us the complete edge set E' without any further computations, since we do not need to compute the overlap quality score for exact overlaps. Note that the minimal overlap length in the contig-overlap graph does not need to be as high as before error correction and it is independent of the read length: all experiments were performed using a minimal contig overlap of 50 bp.

Remark. Allowing approximate overlaps in this stage of the algorithm, for example by allowing some substitutions, would slow down the contig-overlap graph construction considerably. Although the additional edges could lead to improved recovery of true haplotypes, it would also bring the risk of collapsing highly similar sequences and thus missing haplotypes.

3.2.4 Branch reduction in the contig-overlap graph

Before using the contig-overlap graph to extend our contigs, we trim the graph by removing redundant vertices and edges and resolving branches based on read evidence where possible, now also exploiting the paired-end information. After completing this step, we have a 'reduced graph' (see Figure 3.1) that is ready for contig extension.

Transitive edge removal. An edge $u \to w \in E'$ is called *transitive* if there exists a vertex $v \in V'$ and edges $u \to v, v \to w \in E'$. Now that sequences (contigs) are assumed to be error-free, transitive edges have become fully redundant, hence we remove all transitive edges from the graph before further processing.

Branching edges and nodes. The *indegree* (resp. *outdegree*) of a node $v \in V'$ is defined as the total number of incoming (resp. outgoing) edges in G'. If v has indegree greater than one, we say v has an *in-branch*; analogously, if v has outdegree > 1, we say that v has an *out-branch*. We refer to the corresponding edges as *branching edges* and to v as a *branching node*. Since we did not use any read pairing information during construction of our overlap graphs, we observe many branches in the contig-overlap graph. We now use the information how ends are paired to remove any branching edges in the contig-overlap graph that do not correspond to a true haplotype.

Merging simple paths. Following the above definition, any edges that are not branching edges constitute *simple paths* through the contig-overlap graph. For such paths, there is only one possible way to combine the corresponding contigs; hence, before processing the graph any further, we merge every simple path into a single contig. Since edges in the graph represent exact overlaps, this is a straightforward procedure.

Branching components. After merging simple paths, all remaining edges are branching edges. We define a *branching component* as a subgraph H of the contig-overlap graph, such that (1) H is an induced subgraph, (2) H is connected as an undirected graph, and (3) within H, any vertex has only incoming or outgoing edges in H, but not both. A branching component is defined to be maximal with respect to these three properties; see Figure 3.2. Intuitively, a branching component reflects all possible haplotypes within a small region of the genome.

Note that different components may intersect across their vertex sets, but cannot have any edges in common. In other words, the maximal branching components partition the set of all branching edges, as illustrated in Figure 3.2. This partition can be found in time linear in the number of branching edges by alternatingly traversing in-branch edges and out-branch edges until every edge has been seen exactly once; see Section 2 in the Supplementary Material for further details. After enumerating all maximal branching components, we evaluate read evidence per component.

Read evidence. The main idea of read-based branch reduction is to remove all branching edges for which there is insufficient *read evidence* in the input data. For this purpose, we keep track of all original sequencing reads ('subreads') that were used to build a contig; each of these subreads may provide evidence for a branching edge. Within a branching component, we first list all *variant positions*, i.e., the positions at which the sequences corresponding to the different neighbors differ from each other. These are the positions where we may find sequencing reads supporting a given branching edge. A paired-end sequencing read $R = (R_1, R_2)$ is marked as *evidence* for the branching edge $u \to v$ if it satisfies the following conditions:

(i) R spans the branching edge, meaning that at least one of the sequences R_1, R_2 is a subread of u *and* at least one of the sequences R_1, R_2 is a subread of v;

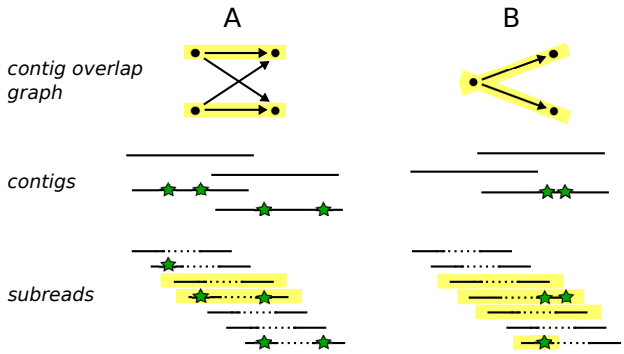

Figure 3.3: Two examples of contigs creating branches in the overlap graph. Edges corresponding to true haplotypes are highlighted in yellow. The corresponding subreads are aligned below, those providing read evidence are again highlighted. (A) Only two out of four edges are supported by read evidence, the other edges will be removed. (B) Both edges are supported by read evidence.

(ii) The sequence spanning the edge is identical to the contig sequence of the corresponding node for all variant positions it covers;

(iii) R is unique for this edge: it does not satisfy conditions (i) and (ii) for any other edge involved in this branching component.

Figure 3.3 shows two examples of contigs creating branches in the overlap graph, along with the sequencing reads ('subreads') that were used to build these contigs; the subreads providing read evidence are highlighted in yellow. Observe that in panel A, in order to satisfy condition (iii) a subread has to cover at least one variant position on either contig. In panel B, we illustrate that also a single read end can provide evidence: the rightmost subread covers a variant position and satisfies all conditions listed above.

Note that condition (ii) ensures that erroneous contigs do not find evidence in correct reads: if a sequencing error accidentally ends up in a contig, it will cause a branch in the overlap graph which can only be supported by reads containing exactly this sequencing error. Whenever such a branch occurs, there will be insufficient evidence and hence the erroneous contigs will never be merged. Eventually, these contigs can be filtered out based on their short length. In the Supplementary Material we discuss how an appropriate evidence threshold is determined (using similar considerations as for determining the optimal clique size, Section 3.2.2). Increasing the evidence threshold would lead to a higher accuracy but also potential loss of low abundance haplotypes.

Branching edge removal. For every branching component, we count the read evidence per branching edge and remove any edges with evidence count below the

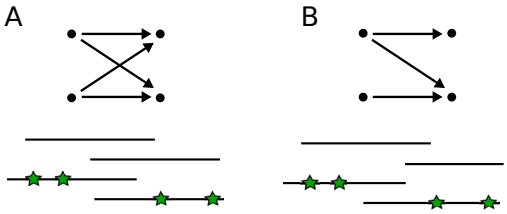

Figure 3.4: Typical branching components in diploid assemblies: four contigs, two from each haplotype, having identical sequence in their overlap. Depending on the contig lengths, all contigs overlap (panel A) or only a subset of the contigs overlap (panel B).

evidence threshold.

3.2.5 Contig extension and graph updating.

After applying the read-based branch reduction techniques described above, all branches have been either resolved or removed from the contig-overlap graph. Contig extension has become an easy task: any contigs which are connected by an edge in the graph must belong to the same haplotype, and, therefore, we merge each such pair of contigs into a new, longer contig. Then, we update the overlap graph: the extended contigs become the new nodes and the edges are updated accordingly. The resulting updated graph is used for further assembly in an iterative manner, as described in Section 3.2.6.

3.2.6 Iterative procedure and diploid mode

Our workflow consists of iteratively performing the steps described in Sections 3.2.3-3.2.5, as illustrated in Figure 3.1. The number of edges in the contig-overlap graph decreases with every iteration, since contigs connected by an edge in the graph are merged (Section 3.2.5). The algorithm terminates when the edge set E' of the updated contig-overlap graph becomes empty, either upon construction or after branch reduction. Thus, our algorithm is guaranteed to converge, and once it does we remove any remaining inclusions from the final contig set. Also any contigs shorter than the fragment size of the original reads are removed from the output.

Diploid mode. Knowing that a given sample is diploid is a very strong piece of information when performing haplotype assembly. We have developed a special module which can be activated for diploid samples. It extends the POLYTE pipeline by two additional steps after the standard algorithm has terminated: construction of a diploid contig graph, followed by contig extension (see Figure 3.1). In these additional steps, we use the knowledge that the sample is diploid to resolve additional branches (for which

there was insufficient evidence in the read set to resolve them during the read-based branch reduction step; see Section 3.2.4).

In overlap graphs from diploid samples we typically see two types of branching components; Figure 3.4 illustrates both types (Panel A and B) and gives an example of a possible collection of contigs giving rise to the corresponding branching component. In both situations we have four contigs, two from each haplotype, which have identical sequence where the contigs overlap. In diploid mode, a single read of evidence may already be considered sufficient, depending on the amount of evidence found for the other edges (Supplementary Material, Section 3).

This procedure is more risky than default branch reduction (Section 3.2.4), since it does not require such stringent read evidence. Therefore, we always run the main POLYTE algorithm until convergence before turning to diploid mode (Figure 3.1). This ensures that all evidence in the original reads has been exploited first.

3.2.7 Software availability

Software and analysis scripts are publicly available as part of the HaploConduct package at https://github.com/HaploConduct/HaploConduct.

3.3 Results

In this section we show results for POLYTE on both simulated and real Illumina data sets and evaluate the assembly quality in terms of haplotype coverage, N50, NGA50, error rate, and the number of misassembled contigs relative to the total number of contigs. We also compare our method against alternative haplotype reconstruction tools: SPAdes [9], Phaser [20], HapCut2 [36], WhatsHap [91], SGA [110], and H-PoP [128]. Other polyploid assemblers [2, 12, 29] were unable to process our benchmark data due to issues with the available software. All methods were run with default settings and assembly statistics were obtained with QUAST [47].

3.3.1 Data sets

Simulated data. We generated a collection of simulated data sets of varying ploidy and sequencing depth to evaluate the effect of these characteristics. We selected four human MHC haplotypes (COX, DBB, MANN, and SSTO) from the Vega Genome Browser [37]. Subsequently, we used SimSeq [112] to simulate Illumina MiSeq reads of length 2×250 bp for each of those haplotypes at a coverage of 5x, 10x, 20x, 30x, 40x, and 50x, respectively, and combined the resulting read sets to form data sets of ploidy 1 (only COX haplotype, a sanity check), ploidy 2 (COX and DBB), ploidy 3 (COX, DBB, and MANN), and ploidy 4 (all).

Real data. For evaluation on real sequencing data, we considered a data set from phase 3 of the 1000 Genomes project [1, 113] for individuals NA19240. This data set was obtained from a 2 × 250 bp PCR free Illumina protocol, sequenced to a coverage of 28-68x. Full haplotypes have been reconstructed for this individual as part of a recent study [21] using various specialized sequencing techniques and reconstruction algorithms; we use the resulting haplotypes as a ground truth for a whole-chromosome benchmark experiment on chromosome 22.

Alignments and variant call sets for reference-guided methods. Reference-guided methods Phaser, HapCut2, WhatsHap, and H-PoP require as input a reference genome, read alignments to the reference genome, and a pre-computed set of genomic variants. For the simulated data we performed read alignment to the GRCh38 reference genome using BWA MEM [65]. The real data were already provided as alignments to the GRCh37 reference genome, also obtained with BWA MEM. We extracted the sequencing reads corresponding to chromosome 22 from the provided BAM files. Finally, we performed variant calling on all data sets with FreeBayes (https://github.com/ekg/freebayes).

3.3.2 Assembly performance criteria

We evaluate assembly performance in terms of several statistics commonly used for de novo assembly evaluation, as reported by QUAST.

Haplotype coverage (HC). The completeness of the assembly is measured by the fraction of nucleotides in the target haplotypes (ground truth) covered by haplotigs, referred to as the haplotype coverage.

N50 and NGA50. Assembly contiguity is measured using the N50 value, which is defined as the length for which the collection of all contigs of that length or longer covers at least half the assembly. The NGA50 measure is computed in a similar fashion, but only aligned blocks are considered (obtained by breaking contigs at misassembly events and removing all unaligned bases). This measure reports the length for which the total size of all aligned blocks of this length or longer equals at least 50% of the total length of the true haplotypes.

Error rate (ER) and N-rate (NR). We evaluate error rate as the sum of mismatch rate and indel rate when comparing to the ground truth haplotype sequences. In addition, we report the relative number of ambiguous bases ('N's), referred to as the N-rate.

Misassembled contigs (MC). A contig or haplotig is called misassembled if it contains at least one misassembly, meaning that left and right flanking sequences align to the true haplotypes with a gap or overlap of more than 1kbp, or align to different strands, or even align to different haplotypes. We report the proportion of misassembled contigs.

	HC (%)	N50	NGA50	ER (%)	NR (%)	MC (%)
Simulated data						
POLYTE	92.4	4397	4394	0.035	0	0
SGA	73.4	3444	-	0.025	0	0
SPAdes	84.1	3588	919	0.032	0	0.0
SPAdes-dip	83.6	3294	903	0.003	0	0
HapCut2	84.5	29259	17980	0.068	0	2.1
H-PoP	81.7	32319	17484	0.158	0	1.7
Phaser	82.6	24785	16884	0.095	0	1.8
WhatsHap	85.2	32656	17980	0.098	0	2.2
Real data						
POLYTE	78.2 *(90.5)*	2838	2316	0.090	0	0.2
SGA	57.7 *(66.8)*	2842	-	0.069	0	0.0
SPAdes	67.0 *(77.5)*	5798	-	0.131	0	0.6
SPAdes-dip	66.4 *(76.9)*	5772	-	0.139	0	0.8
HapCut2	70.1 *(81.1)*	6541	5306	0.090	0.9	0.2
H-PoP	62.4 *(72.2)*	9583	7435	0.119	0.9	0.2
Phaser	66.2 *(76.6)*	6394	5245	0.094	0.9	0.2
WhatsHap	67.6 *(78.2)*	6257	6094	0.092	0.9	0.2

Table 3.1: Benchmarking results, HC = Haplotype Coverage, ER = Error Rate (mismatches + indels), NR = N-Rate (ambiguous bases), MC = Misassembled Contigs. Top: simulated diploid data for the MHC region. Bottom: real data for chromosome 22 of 1000 Genomes individual NA19240. HC values within parentheses indicate haplotype coverage relative to the amount of bases covered by sequencing reads.

3.3.3 Benchmarking results

We performed benchmark experiments on one of the simulated MHC data sets described above (ploidy 2, 20x coverage per haplotype) to compare a variety of haplotype reconstruction tools. In addition, we ran all methods on the chromosome 22 data of the 1000 Genomes individual NA19240. The assembly statistics on both data sets are shown in Table 3.1. Since both data sets are diploid, we present results for SPAdes in regular mode and in diploid mode, referred to as SPAdes-dip.

In both experiments, we observe that across all methods POLYTE has the largest haplotype coverage (HC, 92.4% and 78.2% for MHC and chr22, respectively). In other words, it reconstructs the largest fraction of the true haplotype sequences. In compar-

ison, the other methods are all more or less on a par (81.7–85.2% [MHC] and 57.7–70.1% [Chr22], respectively). On the real data the haplotype coverage achieved by all methods is rather low; this can be explained by only 86.4% of the target haplotypes being covered by sequencing reads. After normalizing the haplotype coverage values by 86.4, POLYTE achieves a haplotype coverage of 90.5%.

In terms of assembly contiguity, indicated by high N50 and NGA50 values, reference-guided methods (HapCut2, Phaser, WhatsHap, H-PoP) perform better than de novo assemblers (POLYTE, SGA, SPAdes). This reflects a common advantage of reference-guided approaches, which can make use of the external information to bridge regions only poorly covered with informative reads, if appropriate. The increase in length, however, is offset by a substantial decrease in terms of haplotig quality: reference-guided approaches exhibit both substantially more misassemblies (which in particular can lead to severe issues in downsteam interpretations) and increased error rates, here larger by one to two orders of magnitude. Note that several NGA50 values are undefined ('-'), because the aligned blocks are unable to cover at least 50% of the total reference length.

Another important difference between reference-guided methods and de novo approaches is reflected in the N-rates on the real data: the reference genome contains several stretches of ambiguous nucleotides ('N's), which the reference-guided methods cannot correct. De novo approaches, on the other hand, can potentially uncover the true sequence behind these ambiguous regions and show an N-rate of 0% (versus 0.9% for the reference-guided methods).

Between de novo approaches, we compare POLYTE with SGA and SPAdes and observe that POLYTE reconstructs a substantially larger fraction of the true haplotypes. Although SPAdes achieves better N50 values, this comes at the expense of a decrease in terms of error rate and misassemblies, also reflected in a low NGA50 value on the simulated data and the NGA50 being undefined on the real data (see explanation above). On the simulated data set, POLYTE and SPAdes achieve comparable error rates of 0.035% and 0.031%, respectively. On the real data we notice an advantage for POLYTE, with an error rate of only 0.090% compared to 0.131% for SPAdes. In addition, POLYTE is less vulnerable to misassemblies than SPAdes on real data, with 0.2% versus 0.6% MC. SGA is able to reconstruct highly accurate contigs with slightly lower error rates than POLYTE (0.025 versus 0.035% [MHC] and 0.069 vs 0.090% [Chr22], respectively), but covers a significantly lower fraction of the ground truth haplotypes (73.4 vs 92.4% [MHC] and 57.7 vs 78.2% [Chr22], respectively).

In an overall account, we believe that, arguably, the major advantage of POLYTE is established by the increase of 10-15% over the other approaches in terms of haplotype-specific coverage, in combination with the error rates, which are clearly lower than those of the other tools.

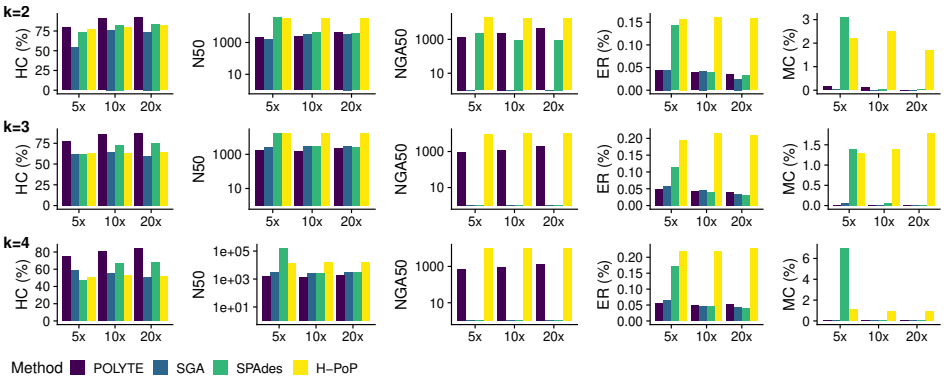

Figure 3.5: Assembly results per method for simulated data of increasing ploidy (k=2,3,4) and per-haplotype coverage (5x,10x,20x). N50 and NGA50 values are plotted on a log-scale for increased readability. For SGA (k=2,3,4) and SPAdes (k=3,4) the NGA50 values are undefined.

In terms of runtime and memory usage, de novo approaches are in general more expensive than reference-guided methods. We also observe this when comparing CPU time and peak memory usage (Supplementary Tables 3–5). Reference-guided methods have CPU times that are orders of magnitude less compared to de novo methods (where POLYTE requires 9–15 times more (resp. 3–6 times more) runtime and 3 times less (resp. 12–17 times more) memory than SPAdes and SGA, respectively). It is important to notice, however, that these de novo assemblers are highly parallelizable; we demonstrate the effect of increasing the number of available CPUs on the effective runtime in Supplementary Table 6. This leads to feasible runtimes on multi-core computing facilities in practice.

3.3.4 Effect of ploidy and sequencing depth

To study the effect of genome ploidy and sequencing depth on the assembly quality and completeness, we ran POLYTE, SPAdes, SGA, and H-PoP on all simulated data sets described in Section 3.3.1 (other tools were unsuitable for polyploid genomes). Figure 3.5 shows the results for the 5x, 10x, and 20x data sets in terms of haplotype coverage (HC), N50, NGA50, error rate (ER), and misassembled contigs (MC). For additional result tables we refer the reader to the Supplementary Tables 8–11.

We observe that POLYTE excels regarding haplotype coverage, with advantages becoming more distinct as the ploidy increases. SPAdes and H-PoP achieve more contiguous assemblies (higher N50 values) but, as we already observed on diploid data, this

comes at the cost of significantly higher error rates and misassemblies. SGA performs very similar to POLYTE when considering N50, ER, and MC, but obtains much lower HC values. The NGA50 values highlight the improved assembly quality of POLYTE over SGA and SPAdes: while POLYTE achieves NGA50 values comparable to the N50, SGA and SPAdes are unable to cover at least 50% of the ground truth with alignments (hence NGA50 is undefined). Overall, we conclude that in polyploid settings the same advantages of POLYTE apply as in diploid settings – increased haplotype-specific coverage in combination with low error rates – and become even more pronounced.

All other methods evaluated (HapCut2, Phaser, Whatshap, SPAdes-dip) are designed for diploid data, so for those we could only assess the effect of sequencing depth. Results indicate that each of the reference-guided methods already performs optimally at a coverage per haplotype of 5x. Moreover, these methods are unaffected by a further increase in sequencing depth (see Supplementary Table 12). SPAdes in diploid mode (SPAdes-dip) performs optimally at a per-haplotype coverage 20x.

3.4 Discussion

Assembling the individual haplotypes of an organism from sequencing reads is known as *haplotype-aware genome assembly* and plays a major role in various disciplines, including genetics and medicine [45, 114]. Computing haplotype-specific pieces of sequence, also known as *haplotigs*, is a difficult task. Algorithms addressing this task do not only need to distinguish between sequencing errors and true variants, but also need to assign the true variants to the individual haplotypes. Enormous quantities of next-generation sequencing (NGS) reads generated worldwide have not been fully exploited in terms of haplotig computation, because methodology for de novo haplotig computation from NGS reads has been in a rather immature state.

We have presented POLYTE (POLYploid genome fitTEr) as a new approach to de novo assembly of haplotigs from NGS data, suitable for diploid genomes as well as genomes of higher ploidy. Unlike the majority of NGS based de novo assemblers, our method follows the overlap-layout-consensus (OLC) paradigm to achieve enhanced performance rates in terms of haplotype-specific computation of contigs. In order to appropriately distinguish between errors and true variants to be assigned to haplotypes, it employs an iterative OLC scheme. Along the iterations, contigs grow in length while preserving their uniqueness in terms of haplotype identity. As a result, POLYTE outperforms the currently available state-of-the-art approaches for haplotig computation, where it performs particularly favorable in terms of quantities that refer to haplotype-specific reconstruction of the genomes.

Experimental results showed that POLYTE can build accurate assemblies from Illumina MiSeq reads (2×250 bp) for data sets of varying ploidy (di-, tri- and tetraploid),

with results for tetraploid organisms almost on a par with those for diploid organisms. Although building overlap graphs for larger genomes remains a challenge, we provide a read binning step that allows efficient assembly by splitting the work over multiple cores. The typical use-case for POLYTE consists of Illumina NGS reads for a specific gene, region or genome that is highly polymorphic (of any ploidy > 1).

We showed that POLYTE succeeds in accurate reconstruction of individual haplotypes of the human MHC region. Future work may therefore be to apply POLYTE to NGS data in population-scale human genome projects, where individual genomes are still lacking proper annotation of their MHC region, which applies in the majority of cases. Advantages become particularly distinct on data of higher ploidy, leading to plant genome assembly as another interesting future application of POLYTE.

Our algorithm is, in its essence, generic in the choice of input reads, so applying it for TGS reads essentially is a matter of adapting parameters, which we will explore in the short-term future.

CHAPTER 4

FULL-LENGTH DE NOVO VIRAL QUASISPECIES ASSEMBLY THROUGH VARIATION GRAPH CONSTRUCTION

In Chapter 2 we took the first steps towards full-length haplotype reconstruction in viral quasispecies. We observed that having a reference-free approach has significant advantages, as reference-induced biases can become overwhelming when dealing with divergent strains. Although we were able to construct strain-specific contigs through de novo assembly, these contigs remained rather short compared to the genome size. This chapter continues our quest for full-length viral quasispecies assembly without a reference genome.

We present Virus-VG, a de novo approach to viral haplotype reconstruction from pre-assembled contigs. Virus-VG is based on the construction of a variation graph from short input contigs. We define a minimization problem that yields a selection of maximal-length paths and the corresponding abundance estimates, which are optimal in terms of being compatible with the read coverages computed for the nodes of the variation graph. Benchmark experiments on challenging simulated and real data sets show significant improvements in assembly contiguity compared to the input contigs, while preserving low error rates compared to state-of-the-art viral quasispecies assemblers.

Published as:
J.A. Baaijens, B. van der Roest, J. Köster, L. Stougie, A. Schönhuth. Full-length de novo viral quasispecies assembly through variation graph construction. *Bioinformatics*, btz443, 2019 (in press).

Supplementary material: https://doi.org/10.1093/bioinformatics/btz443.

4.1 Introduction

The ensembles of genetically related, but different mutant viral strains that populate infected people are commonly referred to as *viral quasispecies* [32]. Each of these strains comes with its own genomic sequence (henceforth referred to as *haplotype*). The final goal of primary viral quasispecies analysis is the reconstruction of the individual haplotypes—optimally at full length—and also to provide estimates of their abundances. The unknown number of different, strain-specific haplotypes and their variance in abundance establish the theoretical issues that characterize viral quasispecies assembly. They explain why this form of assembly is difficult, despite the shortness of virus genomes. These issues are further accentuated by the fact that neither next-generation nor third-generation sequencing reads, by their combinations of error rates and length, allow for immediate reconstruction and abundance estimation of haplotypes [11, 102].

State-of-the-art approaches currently allow for two options: **(i)** full-length reconstruction of haplotypes based on statistical, usually *reference genome dependent* measures, or **(ii)** *de novo* reconstruction of (optimally haplotype-specific) contigs.

Approaches of type **(i)** assume that the sequencing reads are aligned to a reference genome and make use of model-based clustering algorithms [3, 10, 130], Dirichlet process mixture models [97], hidden Markov models [119], sampling schemes [98], or combinatorial methods [57], respectively. However, as was demonstrated in [7, 118], resorting to external auxiliary means (such as reference genomes) can bias the reconstruction procedure significantly.

Approaches of type **(ii)** comprise generic (meta)genome assemblers as well as specialized viral quasispecies assemblers, both of which are not helped by external measures (*"de novo"*) hence are not affected by external biases. Metagenome assemblers are designed to reconstruct multiple genomes simultaneously, but in viral quasispecies tend to collapse strains [102]. It was further shown by Baaijens et al. [7] that among generic de novo assemblers SPAdes [9] was the only approach to identify strain-specific sequences, however only in case of sufficiently abundant strains. *De novo* viral quasispecies assemblers (e.g. [51, 129]) generally aim at constructing suitable consensus reference genomes, which may serve as a template for more finegrained studies (for example if curated reference genomes have become too divergent, which is a frequent scenario). The only methods that truly aim at *de novo* genome assembly *at strain resolution* are SAVAGE [7], MLEHaplo [72] and PEHaplo [22]. However, the contigs produced by these methods, while strain-specific, in general do not represent full-length haplotypes.

We present Virus-VG, an algorithm that turns strain-specific contigs into full-length, strain-specific haplotypes, thus completing the de novo viral quasispecies assembly task. For that, we construct a variation graph from the contigs, without the help of a (curated) reference genome, where we use the contigs produced by SAVAGE [7]. We

obtain full-length haplotypes as a selection of maximal-length paths in the variation graph, each of which reflects a concatenation of subpaths associated with the input contigs. The selected paths are optimal in terms of differences between their estimated abundances and the read coverages computed for the nodes they traverse. Although our approach to quasispecies assembly using candidate path enumeration followed by abundance estimation is similar to Astrovskaya et al. [6] and Skums et al. [111], these methods make use of read graphs rather than variation graphs and define optimality for path abundance estimation in a very different way.

Variation graphs are mathematical objects that have recently become very popular as reference systems for (evolutionarily coherent) collections of genomes [90]. Using such genome structures instead of standard linear reference genomes has been shown to reduce reference bias [31, 90] and to allow for efficient subhaplotype match queries [88] and haplotype modelling [103]. Methods presented so far for constructing variation graphs, however, have been focused on a linear reference genome as a point of departure. Here, we point out how to construct variation graphs *de novo*, by first sorting the contigs in an appropriate way and then making use of progressive multiple alignment techniques (`vg msga`, part of the `vg toolkit` by Garrison et al. [43]). In this, we present an approach for full-length, high-quality reconstruction of the haplotypes of a viral quasispecies that is *entirely de novo*.

Our method depends on the enumeration of maximal-length paths in a variation graph, whose number is in the worst case exponential in the number of nodes of the graph. However, since all these paths enumerated are to respect the subpaths associated with the input contigs, their number will decrease on increasing contig length. Thanks to advances in sequencing technology, input contig length will inevitably increase, which points out that our method, as per its design, will be able to deal with future technological developments smoothly.

Benchmark experiments demonstrate that Virus-VG yields substantial improvements over the input contigs assembled with SAVAGE in terms of spanning the full length of the haplotypes. Thereby, the increase in length comes at negligible or even no losses in terms of sequential accuracy compared to the input contigs. Further, we find our strain abundance estimates to be highly accurate. Finally, we find our method to (substantially) outperform alternative approaches, all of which are reference based—we recall that there are no alternative de novo approaches so far—both when working with bootstrap (i.e. assembled from the data itself) and curated reference genomes.

Note on Related Work: RNA Transcript Assembly. Many RNA transcript assemblers work in a similar way to Virus-VG: first enumerating all possible transcripts, then selecting an 'optimal' set of transcripts using various optimization methods [40, 67, 81]. This has led to variations of the minimum path cover optimization problem that are—regarding a few relevant aspects—similar in spirit to the optimization problem we

formulate [13, 93, 101, 117, 120]. Most importantly, [101] introduce node and edge abundance errors and [117] show a minimum path cover with subpath constraints to be polynomially solvable. However, to the best of our knowledge, no method simultaneously employs both subpath constraints *and* abundance error minimization in its problem formulation. Moreover, applying these RNA transcript assemblers to the viral quasispecies problem is not so straightforward: a collection of reference genomes representing all possible haplotypes is required as input, while in our setting such information is not available.

4.2 Methods

Notation. A *variation graph* (V, E, P) is a directed graph that is constructed from a set of input sequences, which represent members of a (evolutionarily coherent) population of sequences. Each node $v \in V$ is assigned to a subsequence $\text{seq}(v)$. An edge $(u, v) \in E$ indicates that the concatenation $\text{seq}(u)\text{seq}(v)$ is part of one of the input sequences. P is a set of paths (a sequence of nodes linked by edges) that represent genome-specific sequences; thereby, P can, but need not, represent the input sequences themselves. A node $v \in V$ with no incoming edges is called a *source*. A node $v \in V$ with no outgoing edges is called a *sink*.

Workflow. Our method consists of two basic steps:

- **(1)** The computation of a ***contig-variation graph*** $VG' = (V', E', P')$ where each path $p \in P'$ represents an input contig. We refer to the path representing contig c as $p(c)$. Together with VG', we compute a function $a' : V' \to \mathbb{R}$ where $a'(v')$ for $v' \in V'$ represents the abundance of an individual node, measured by the amount of original reads (from which the contigs were computed) that align to $\text{seq}(v')$.
- **(2)** The transformation of VG' into a ***genome-variation graph*** $VG = (V, E, P)$ where each path $p \in P$ reflects a full-length haplotype. We also compute a function $a : P \to \mathbb{R}$ where $a(p)$ for $p \in P$ reflects the abundance of the haplotype represented by p. The set of paths P together with their abundances $a(p)$ establish the final output of our method.

The input for determining VG' in **(1)** are the contigs. For computation of a', we make use of the original reads from which the input contigs were computed; one can determine the abundance $a'(v')$ of single node $v' \in V'$ as the (length normalized) count of reads whose alignments touch upon v'.

The input for computation of VG and a in **(2)** are VG' and a'. Since $V \subseteq V'$ and $E \subseteq E'$, as will become clear later, we can apply a' also to nodes in VG. The computation of VG is established as the solution of an optimization problem that aims to determine full-length paths (paths formed by a concatenation of contigs of maximal length) such

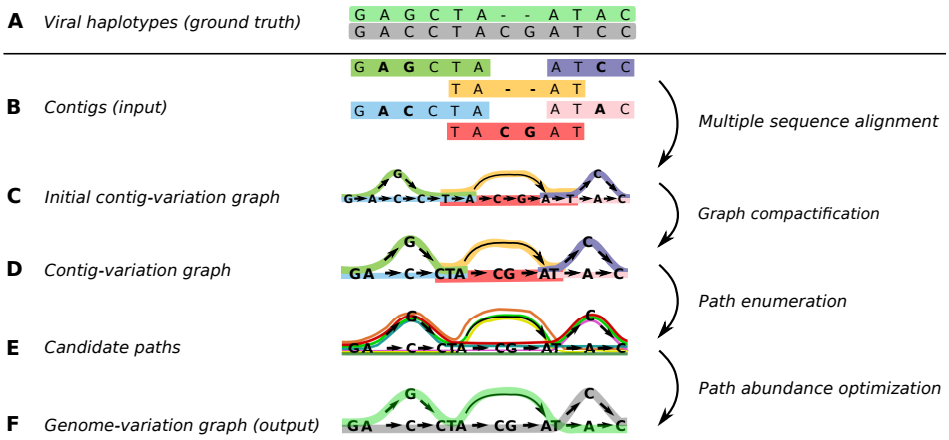

Figure 4.1: Virus-VG workflow. (A) Ground truth haplotypes from which the sequencing reads originate; (B) Input contigs, obtained by de novo assembly; (C) Initial contig-variation graph created from input contigs using multiple sequence alignment; (D) Contig-variation graph obtained by collapsing non-branching paths (compactification); (E) Candidate paths representing possible haplotypes; (F) Genome-variation graph representing the haplotypes selected through path abundance optimization, capturing the viral quasispecies.

that the difference of path abundances $a(p)$ and node abundances $a'(v)$ for paths p of which v makes part is minimal. We emphasize here that the numbers $a'(v)$ can be directly computed from the input, whereas the $a(p)$'s correspond to decision variables in an optimization problem.

We will describe the construction of the contig-variation graph **(1)** in full detail in Section 4.2.1. The transformation into the (final) genome-variation graph **(2)** is divided into two steps: **(a)** the *enumeration of candidate paths*, which is described in Section 4.2.2, and **(b)** the *solution of an optimization problem* that aims at selecting a subset of candidate paths through their path abundance values which are optimal in terms of being compatible with node abundances in Section 4.2.2. The complete workflow is illustrated in Figure 4.1.

4.2.1 Contig-variation graph construction

Input. The input is a data set of next-generation sequencing reads and a set of contigs assembled from them, for which we use the specialized de novo viral quasispecies tool SAVAGE [7]. We assume that there are no contigs which are an exact subsequence of

another contig, which applies for SAVAGE (and commonly applies for the output of many assembly programs); any such contigs are removed. The contig-variation graph with its node abundances is constructed in three steps.

Step 1: Multiple Sequence Alignment (MSA). We construct the initial contig-variation graph by building an MSA of the contigs using `vg msga` [43], which progressively combines long sequences into a variation graph. For this construction to work on a collection of contigs that do not all cover the same region, the order in which the contigs are aligned and added to the graph is important: we need to add contigs such that they overlap the existing graph as much as possible, to avoid creating more disjoint components than necessary. Here, we sort the contigs by starting with the longest contig, then iteratively selecting the contig with longest possible overlap with any of the previously selected contigs, until all contigs have been selected. For finding all pairwise overlaps between contigs we use `minimap2` [66]. Determining the best sorting heuristic in terms of speed and quality is subject to future work. After sorting the contigs, we apply `vg msga`; the resulting MSA is represented as a variation graph and for every contig the corresponding path through the graph is stored.

Step 2: Compactification and contig-path construction. We compactify the initial contig-variation graph in a similar fashion as in the construction of a compacted de Bruijn graph [70]. The absence of branches on a path ensures that every source-sink path has to traverse it at full length. Therefore, each non-branching path $(v_{i_1}, \ldots, v_{i_k})$ can be merged (contracted) into a single node v'_i, with in-neighbors $N^-(v'_i) = N^-(v_{i_1})$ and out-neighbors $N^+(v'_i) = N^+(v_{i_k})$. Also the contributing contig sets of $v_{i_1}, \ldots, v_{i_k}$ are taken together and stored in the new node v'_i. Note that after this step, a node can represent a sequence instead of a single nucleotide.

In addition, we determine for each contig c the sub-path $p(c)$ in this (compacted) graph that represents c. Let $p(c) = (v_{i_1}, \ldots, v_{i_k})$ be this sub-path. Note that due to the compression step, the sequence $\mathrm{seq}(c)$ represented by a contig c might only be a subsequence of its path sequence $\mathrm{seq}(v_{i_1})$...$\mathrm{seq}(v_{i_k})$. However, this does not bear any consequence on the definition of any haplotype the contigs make part of.

The resulting compacted graph, together with the contig paths P' is our contig-variation graph $VG' = (V', E', P')$, illustrated in Figure 4.1, panel D.

Step 3: Node abundance. We finally compute $a' : V' \to \mathbb{R}$, which assigns *node abundances* $a'(v')$ to nodes $v' \in V'$ of the contig-variation graph. These node abundances $a'(v')$ reflect the *average base coverage* of the piece of sequence $\mathrm{seq}(v')$. For computation of $a'(v')$ we make further use of the `vg-toolkit` [43], which allows to align the original

sequencing reads to our contig-variation graph. The abundance $a'(v')$ is calculated as the sum of all bases in all reads that align with $\text{seq}(v')$, divided by the length of $\text{seq}(v')$.

4.2.2 From contig to genome-variation graph

The input for the following procedure is the contig-variation graph $VG' = (V', E', P')$ together with $a' : V' \to \mathbb{R}$ that we have just described. The procedure for constructing the *genome-variation graph* $VG = (V, E, P)$ from VG' and a' consists of three steps. First, we compute a set of *candidate paths*, which are all maximal-length paths in (V', E') that are "concatenations" of paths from P'. Second, we select a subset of candidate paths that are optimal with respect to a *minimization problem*, which provides us with the final, maximal-length paths P and path abundances $a : P \to \mathbb{R}$. Third, we remove nodes and edges from (V', E') that are not traversed by paths from P, which yields the final graph (V, E). Since only paths in P are supposed to reflect true haplotypes, any node not being included in a haplotype is most likely a sequencing artifact or an assembly error (or it belongs to a missing strain). The third step is a straightforward procedure. We will describe the first two steps in more detail in the following.

Candidate path generation.

The goal is to compute the set of all paths through (V', E') that are maximal-length *concatenations* of paths from P', where we understand a concatenation of two paths as the merging of them along a common substring. Thereby, this common substring is a suffix of the first path and a prefix of the second path. We will refer to these paths as *candidate paths* P_{cand} in the following (see also Figure 4.1, Panel E). Generating candidate paths happens in five steps outlined below.

Step 1: Trimming paths $p \in P'$. Due to common issues in contig computation, uncorrected sequencing errors are often located on the extremities of the contig. We therefore shorten all paths $p \in P'$ by their extremities and remove the tails if these contain nodes v' for which $a'(v')$ is below a given threshold. By default, we allow to trim paths $p \in P'$ by a removal of nodes that together amount to no more than $\tau = 10$ bp on either end.

Step 2: Enumerating pairwise concatenations. We allow concatenating pairs of paths with matching suffix-prefix pairs. In more detail, let $p_1, p_2 \in P'$, represented by series of nodes $(u_1, ..., u_m)$ and $(v_1, ..., v_n)$ from V'. Then p_1 can be concatenated with p_2, written $p_1 \to_c p_2$, if for some l we have $u_{m-l+1} = v_1, u_{m-l+2} = v_2, ..., u_m = v_l$, that is, the suffix of length l of p_1 matches the prefix of length l of p_2.

In order to enable correction of persisting sequencing errors, we further consider to concatenate pairs of paths p_1, p_2 which do have one or more non-matching nodes, but

only under the following condition. Let $u^* := u_{m-l+i} \neq v_i =: v^*$ be the respective non-matching nodes in p_1, p_2 respectively, then only if $\min\{a'(u^*), a'(v^*)\} < \alpha$, where α is a user-defined threshold, we concatenate p_1 and p_2. This threshold reflects the minimal node abundance $a'(v)$ for which we trust node v; for more details, see Appendix A.

Step 3: Removing concatenations lacking physical evidence. Subsequently, we remove concatenations $p_1 \rightarrow_c p_2$ if there are q_1, q_2 such that $q_1 \rightarrow_c q_2, q_1 \rightarrow_c p_2, q_2 \rightarrow_c p_2$, but there is no q_3 for which $p_1 \rightarrow_c q_3$ and $q_3 \rightarrow_c p_2$ and there is q_4 such that $p_1 \rightarrow_c q_4$. The situation reflects that the concatenation of paths $q_1 \rightarrow_c p_2$ enjoys corroborating physical evidence, provided by q_2, while there is no such corroborating evidence for the concatenation $p_1 \rightarrow_c p_2$. At the same time, p_1 concatenates well with q_4 such that the removal of $p_1 \rightarrow_c p_2$ does not turn p_1 into a dead end.

Step 4: Enumerating maximal-length paths P_{cand}. In this step, the pairwise concatenations from step 2 that remain after step 3 are combined to paths of maximal length. This is achieved through a breadth-first search type procedure. We maintain a set of *active paths* P_{act}, which is the set of paths to be extended in the current iteration. We also maintain a set of *maximal paths* P_{max} that reflects the set of maximal-length paths collected.

- **Initialization:** We determine all $p \in P'$ for which there are no $q \rightarrow_c p$ and put them both into P_{act} and P_{max}.
- **Iteration:** We replace each $p \in P_{\text{act}}$ with all $q \in P'$, for which $p \rightarrow_c q$ without q^* such that $p \rightarrow_c q^* \rightarrow_c q$. Simultaneously, we extend each $\hat{p} \in P_{\text{max}}$ that ends in p, by appending q (while respecting the overlap). In case q is already part of $\hat{p}$, we do not append q to $\hat{p}$ but instead add q as a new path to P_{max}, thereby breaking any possible loops due to repetitive elements.
- **Output:** If for all $p \in P_{\text{act}}$ there are no q with $p \rightarrow_c q$, we output P_{max} as our candidate path set P_{cand}.

The enumeration algorithm lists all candidate paths in time linear in the output size. This can be (in the worst case) exponential in the number of paths $p \in P'$, depending on the structure of the contig-variation graph.

Step 5: Correcting paths for errors. After enumerating all candidate paths, we apply a final correction step to every such path. Since we allow concatenating paths from P' where suffix-prefix pairs do not match in all nodes (see Step 2), we may have positions in candidate paths where contig paths $p \in P'$ do not agree on the underlying sequence. All such ambiguous positions refer (by construction) to low abundance nodes v' (i.e.,

$a'(v') < \alpha$). We resolve the ambiguity by selecting the node v^* from all contributing paths $p \in P'$ with maximal abundance $a'(v^*)$.

Minimization for haplotype selection and abundance estimation

Input. For this final part of the method, the input is the set of candidate haplotype paths P_{cand} and the node abundances $a'(v)$. In general this set of paths is much larger than the actual number of haplotypes, so P_{cand} will contain many false haplotypes. Here we filter them out by estimating the abundance $a(p)$ for each path (haplotype) $p \in P_{\text{cand}}$ through solving a minimization problem. In a subsequent step, haplotype paths with an abundance below a user defined threshold will be removed as being most likely false haplotypes. This leaves the set of haplotypes to be output.

Determining path abundances $a(p)$. We determine path abundance values $a(p)$ for every $p \in P_{\text{cand}}$, such as to minimize the sum of or, equivalently, the average of node abundance errors. Let $f(x, y)$ be an error function to be chosen later. Then for node v the node abundance error is defined as the value of $f(x, y)$ with x the node abundance $a'(v)$ and y the sum of the abundances of the haplotype paths going through the node v, which is $\sum_{p \ni v} a(p)$. Recall that the node abundance values $a'(v)$ are obtained from read alignments to the contig-variation graph (Section 4.2.1, Step 3). The objective then becomes minimizing the sum of the node abundance errors over all nodes $v \in V'$:

$$\min \sum_{v \in V'} f\left(a'(v), \sum_{p \ni v} a(p)\right).$$

We need to add non-negativity constraints $a(p) \geq 0$ on the path abundances. Since we have already taken all subpath constraints into account when enumerating the candidate haplotype paths, the minimization problem does not need any further constraints.

Note that the effectiveness of this objective function depends heavily on the error function used as well as the correctness of node abundances $a'(v)$. These abundance values are not exact measurements, but based on read alignments to the graph as described above; coverage fluctuations can thus lead to under- or overestimated node abundance values. In this case, a simple linear objective function is preferred over a quadratic error function, because the former allows big errors in certain nodes to be compensated by small errors in other nodes. We also observed that normalizing the errors w.r.t. the true node abundance does not improve results, because this means that errors in nodes with low abundance values are penalized very strongly. For this reason, we use the error function $f(x, y) = |x - y|$ in our objective and the optimization problem becomes

$$\min \sum_{v \in V'} \left| a'(v) - \sum_{p \ni v} a(p) \right| \qquad \text{s.t. } 0 \leq a(p) \quad \forall\, p \in P_{\text{cand}}. \tag{4.1}$$

This is a convex programming formulation, which can, by a standard transformation, easily be linearized and solved using an LP solver.

Output: haplotype selection and final abundances. The outcome of the minimization problem (4.1) yields for each $p \in P_{\text{cand}}$ an optimal abundance value $a^*(p)$. We now select the set of haplotype paths as output of the procedure, by removing any haplotypes with an estimated abundance below a user defined threshold γ. In other words, as output we give the set $P = \{p \in P_{\text{cand}} \mid a^*(p) \geq \gamma\}$ (Figure 4.1, panel F). After this haplotype selection step, we redo the optimization step on the selected haplotype paths (prefixing $a(p)$ to 0 for every path p with $a^*(p) < \gamma$), thus ensuring that our final abundance estimates are as accurate as possible.

Note on related work. The minimization problem we are treating here can be considered as a combination of problems presented in [101] and [117]. The combination of these problems would require an unambiguous way to have subpath abundances contribute to cumulative abundances on the nodes. It is not immediately evident how to do so. In our setting it is straightforward how path abundances $a(p)$ contribute to the estimated abundances of the nodes on the paths. Exploring these relationships is interesting future work.

4.2.3 Software availability

Software and analysis scripts are publicly available at https://bitbucket.org/jbaaijens/virus-vg.

4.3 Results

We present results for Virus-VG on four challenging simulated data sets and one real benchmark. We compare our method with the viral quasispecies assemblers ShoRAH [130] and PredictHaplo [97], which are widely approved and state-of-the-art in terms of full-length reconstruction of viral haplotypes. On a shorter region (HIV pol gene) we also compare our method to aBayesQR [3] and PEHaplo [22]. Although a comparison to the RNA transcript assemblers from Rizzi et al. [101] and Tomescu et al. [117] would be interesting, this is not so straightforward: these methods require as input a collection of reference genomes representing all possible transcripts (or in our case, viral haplotypes). Since we do not have such information, we could not apply these methods to our data.

For shell commands, parameters to be set, their default choices, and further reasoning, see the Supplementary Material.

4.3.1 Data sets

For evaluating correctness of our algorithm and benchmark experiments on full-length viral genomes, we selected the two most challenging simulated data sets (HCV, ZIKV) presented by [7] and generated one additional data set (Poliovirus). These data sets represent typical viral quasispecies ultra-deep sequencing data and consist of 2×250 bp Illumina MiSeq reads which were simulated using SimSeq [112]. In addition, we simulated mixtures of HIV strains, considering only the 3kb pol region, at various sequencing depths. Finally, we also consider a real Illumina MiSeq data set commonly used for benchmarking, referred to as the *labmix*. More details about all data sets are presented in the Supplementary Material.

10-strain HCV mixture. This is a mixture of 10 strains of Hepatitis C Virus (HCV), subtype 1a, with a total sequencing depth of approximately 20,000x (i.e. 400,000 reads). The haplotypes were obtained from true HCV genomes in the NCBI nucleotide database and have a pairwise divergence varying from 6% to 9%. Paired-end reads were simulated at relative frequencies between 5% and 13% per haplotype, i.e., a sequencing depth of 1000x to 4600x per haplotype.

15-strain ZIKV mixture. This is a mixture of 15 strains of Zika Virus (ZIKV), consisting of 3 master strains extracted from the NCBI nucleotide database and 4 mutants per master strain. The pairwise divergence varies between 1% and 12% and the reads were simulated at relative frequencies varying from 2% to 13.3%. The total sequencing depth for this data set is again 20,000x.

6-strain Poliovirus mixture. This is a mixture of 6 strains of Poliovirus (type 2), with a total sequencing depth of approximately 20,000x. The haplotypes were obtained from true Poliovirus genomes from the NCBI database. Paired-end reads were simulated at exponentially increasing relative frequencies of 1.6% to 50.8%.

7-strain HIV mixture (pol gene). This is a mixture of 7 HIV-1 pol gene sequences with a pairwise divergence of 1%, obtained by introducing random mutations into sequence D86068.1 from the NCBI database. Paired-end reads (2×250 bp) were simulated for each of the 7 strains at relative frequencies between 0.5% and 61.5%. We created 3 data sets with sequencing depths of 500x, 1000x, and 5000x, respectively.

Labmix. This is a real Illumina MiSeq (2×250 bp) data set with an average coverage of 20,000x, sequenced from a mixture of five known HIV-1 strains (HXB2, NL4-3, 89.6, YU2, JRCSF) with relative strain frequencies between 10% and 30%. This data set was presented as a benchmark by Di Giallonardo et al. [30] and is publicly available at https://github.com/cbg-ethz/5-virus-mix. Currently, predictions of all methods, including our own, are hampered by highly repetitive regions such as the long terminal repeats on the HIV genome; see also Baaijens et al. [7]. Hence, we decided to follow [30] in removing these a priori by excluding any reads that map to these known repeat

sequences.

4.3.2 Assembly evaluation criteria

We use QUAST [47] for evaluating our experiments and report the number of contigs, the fraction of the target genomes that was reconstructed, the N50 and NGA50 measures, and observed error rates. Here, the target genome consists of all true haplotypes known to be present in a sample; a base is considered reconstructed if there is at least one contig with an alignment to this base. The N50 measure, defined as the length for which the collection of all contigs of that length or longer covers at least half the assembly, gives an indication of assembly contiguity. The NGA50 measure is computed in a similar fashion, but only aligned blocks are considered (obtained by breaking contigs at misassembly events and removing all unaligned bases). This measure reports the length for which the total size of all aligned blocks of this length or longer equals at least 50% of the total length of the true haplotypes; the NGA50 value is undefined if a target coverage of 50% cannot be reached. Finally, the error rates we present are computed as the sum of the N-rate (i.e. ambiguous bases) and mismatch- and indel rates (compared to the ground truth), normalized by the number of assembled bases. Further details are presented in the Supplementary Material.

4.3.3 Improvements of final haplotypes over input contigs

Table 4.1 presents assembly statistics for all methods on all benchmark data sets. The first two rows, SAVAGE and Virus-VG, display the statistics for the input contigs and the final, maximal-length haplotypes computed here, respectively, for the HCV data set. While SAVAGE presents 26 fragmented contigs, Virus-VG presents 10 full-length haplotypes, each of which represents one of the original haplotypes, thereby encompassing the 10 original haplotypes that established the basis for simulating reads. Further, Virus-VG covers 99.3% of the target genomes, similar to the original 99.4% provided by the input contigs, and these full-length haplotypes come at a negligible error rate of 0.001%. In summary, our approach yields near-perfect results on this (supposed to be challenging) data set.

For the 15-strain ZIKV data set we again achieve substantial improvements in terms of haplotype assembly contiguity. We obtain 20 full-length haplotypes covering 14 out of 15 strains, while the original input contigs consisted of 89 highly fragmented, and relatively short sequences. As a result, we observe an NGA50 value of 10210 for Virus-VG, reflecting full-length haplotypes, compared to an N50 of 3801 for SAVAGE. For the 6-strain Poliovirus mixture we obtain similar results, yielding a major improvement of NGA50 values (1643 for SAVAGE compared to 7428 for Virus-VG) at the cost of a minor

decrease in target haplotypes reconstructed (83.7% for SAVAGE compared to 80.7% for Virus-VG).

On both the ZIKV and Poliovirus data, we observe a slight increase in error rate after applying our method; however, Virus-VG leaves with an error rate of 0.115% (ZIKV) and 0.064% (Poliovirus), which is still extremely low. A thorough analysis turns up that this increase is due to errors in the input contigs that become more expressed only after having assembled the full-length haplotypes, so these errors are not primarily due to the method presented here. Moreover, the full-length contiguity of the haplotypes clearly offsets the minute shift in accuracy.

Finally, we analyze performance on a real benchmark, the labmix, and observe the same behaviour for Virus-VG: a significant improvement in NGA50 values (1450 for SAVAGE compared to 4642 for Virus-VG) but also an increase in error rate (0.066 for SAVAGE compared to 0.324 for Virus-VG). However, it is important to realize that the true sequences considered here may not fully represent the sample, because extremely high mutation rates allow the virus to mutate and recombine in vitro before sequencing.

4.3.4 Comparison with the state-of-the-art

Rows 3 and 4 for every data set in Table 4.1 display results for state-of-the-art methods PredictHaplo [97] and ShoRAH [130], run with default parameter settings. Both of these methods are reference-guided, hence cannot immediately be compared with Virus-VG, which operates entirely de novo. To simulate a de novo type scenario for these reference-guided approaches, we provided them with a bootstrap reference genome computed by running [129, VICUNA], a state-of-the-art tool for generating consensus virus genomes, on the input reads. We also ran [3, aBayesQR] and [22, PEHaplo], but found them unsuitable for reconstructing full-length genomes at ultra deep coverage: they could not finish the job within 500 hours. Hence, we only present results for these methods on the HIV pol region data sets (both simulated and real) in Section 4.3.5.

We first evaluated both PredictHaplo and ShoRAH on our simulated data and, in all cases, we found our method to have (quite significant) advantages, in terms of accuracy, number of strains, and strain-specific genomes covered. As was already observed earlier [7], reference-guided methods greatly depend on the quality of the reference genome provided and have to deal with biases towards the reference genome. This results in error rates which are 1.1–59 times higher than Virus-VG for PredictHaplo, and more than 12 times higher than Virus-VG for ShoRAH. At the same time, these methods miss a big fraction of the target haplotypes on all data sets except the labmix.

PredictHaplo and ShoRAH both had difficulty processing the Poliovirus data. A possible explanation is the high divergence between the virus strains and the reference genome used, leading to gaps in coverage when considering alignments to the reference

	# contigs*	target (%)	N50	NGA50	error rate (%)
10-strain HCV mix					
SAVAGE	26	99.4	8964	8964	0.001
Virus-VG	10	99.3	9281	9203	0.001
PredictHaplo	9	73.8	7636	7608	0.059
ShoRAH	639	56.9	7570	7570	4.294
15-strain ZIKV mix					
SAVAGE	100	98.8	2954	3801	0.023
Virus-VG	20	92.8	10202	10210	0.115
PredictHaplo	8	53.3	10270	10267	0.126
ShoRAH	493	26.3	10117	10117	4.392
6-strain Poliovirus mix					
SAVAGE	59	83.7	1089	1643	0.019
Virus-VG	14	80.7	7316	7428	0.064
PredictHaplo	3	16.6	7461	-	1.825
5-strain HIV labmix					
SAVAGE	68	97.9	1026	1450	0.066
Virus-VG	23	90.6	2130	4642	0.324
PredictHaplo	6	100.0	8825	8825	1.066
ShoRAH	250	100.0	8775	8775	3.910

Table 4.1: Assembly results per data set. Error rates are computed as the sum of the fraction of 'N's (ambiguous bases) and the mismatch- and indel rates. ShoRAH could not process the Poliovirus data. *If contigs are full-length, this number reflects the estimated number of strains in the quasispecies.

genome, which tends to confuse reference-guided methods. In particular, two of the six strains have a big deletion (more than 1000 bp) compared to both the reference genome and the other four strains; this may also explain the failure to run ShoRAH even using a bootstrap reference genome, as well as the extremely low target reconstructed for PredictHaplo. These results again highlight the advantage of a fully de novo approach compared to reference-guided methods.

4.3.5 Gene sequence reconstruction

Although our main goal is to reconstruct full-length genomes, some studies also require sequence reconstruction from data corresponding to shorter regions of the genome. Therefore, we explored the ability of Virus-VG to reconstruct the HIV pol gene sequence,

	# contigs*	target (%)	N50	NGA50	error rate (%)
Coverage = 500x					
SAVAGE	7	51.0	2291	930	0.005
Virus-VG	4	52.7	2974	2330	0.057
aBayesQR	6	55.7	3069	3069	0.249
PEHaplo	64	55.7	3048	3062	0.459
PredictHaplo	1	14.3	3068	-	0.392
ShoRAH	27	61.0	2680	2680	1.322
Coverage = 1000x					
SAVAGE	12	57.2	1416	986	0.012
Virus-VG	6	61.3	2977	2907	0.116
aBayesQR	6	62.8	3070	3070	0.258
PEHaplo	59	61.7	3045	3063	0.449
PredictHaplo	2	21.4	3070	-	0.514
ShoRAH	27	59.8	2680	2680	1.304
Coverage = 5000x					
SAVAGE	17	66.4	1612	1596	0.005
Virus-VG	7	64.7	2853	2864	0.089
aBayesQR	7	61.4	3074	3074	0.283
PEHaplo	469	97.0	3045	3072	0.519
PredictHaplo	2	28.5	3070	-	0.587
ShoRAH	32	51.4	2766	2766	1.404

Table 4.2: Assembly results for the simulated HIV pol region (~3kb) at coverage 500x, 1000x and 5000x. Error rates are computed as the sum of the fraction of 'N's (ambiguous bases) and the mismatch- and indel rates. *If contigs are full-length, this number reflects the estimated number of strains in the quasispecies.

a 3kb region coding for DNA polymerase. This also allowed us to compare our assemblies against aBayesQR[3] and PEHaplo [22], as these methods were unable to process full-length genomes at deep coverage.

Table 4.2 presents results for all methods on simulated data, the 7-strain HIV pol mixture, at sequencing depths of 500x, 1000x, and 5000x. This data set is very challenging, with pairwise divergence of only 1% and relative abundances ranging from 0.5% and 61.5%. Therefore, it is not surprising to see relatively low target reconstructed for all methods; the only method that is able to find the strain of 0.5% abundance is PEHaplo at 5000x coverage (see Supplementary Material). For all methods except ShoRAH, reconstructed target values improve upon increasing coverage. Virus-VG shows similar behaviour as on full-length genomes: it gives a major improvement in N50 and NGA50 values compared to SAVAGE, and while error rates are higher than for SAVAGE they remain much lower than for other methods.

In addition, we selected reads from the pol region of the labmix (real data) and subsampled this selection to 100x and 1000x coverage, respectively. Again, Virus-VG achieves low error rates in combination with high N50/NGA50 values and improves as sequencing depth increases. To ensure robustness, simulations and subsampling were performed 10 times, methods were run on these 10 samples and results were averaged.

4.3.6 Haplotype abundance estimation

We also evaluated the accuracy of the abundance estimates obtained for each haplotype of the simulated data sets, since we know the exact true frequencies for each of the strains. The reconstructed sequences were aligned to the ground truth sequences and assigned to the closest matching strain. For each ground truth strain, we summed the abundance estimates of the sequences assigned to it, thus obtaining a total estimate for this strain. Then we compared this estimate to the true strain abundance and computed the absolute frequency estimation errors. In case of any missing strains, the true frequencies were normalized first, taking only the assembled sequences into account for a fair comparison. SAVAGE and PEHaplo assemblies are not evaluated as these methods do not provide abundance estimates.

Our method predicts highly accurate abundances for the reconstructed strains, with an average absolute estimation error of 0.1% on the HCV data, 0.3% on the ZIKV data, and 0.6% on the Poliovirus data. In comparison, PredictHaplo achieves an average absolute estimation error of 0.9% (HCV), 4.9% (ZIKV), and 10.6% (Polio), while ShoRAH is even further off with 8.5% (HCV) and 39% (ZIKV). On the HIV pol data, we observe higher estimation errors for all methods, with an absolute error of 2.6–5.8% for Virus-VG, 3.9–4.9% for aBayesQR, 8.2–22.1% for ShoRAH, and 6.4% for PredictHaplo (only evaluated at 5000x). Relative estimation errors show a similar pattern. A likely explanation

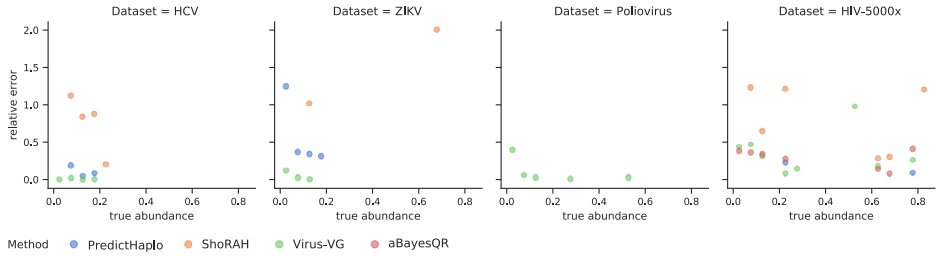

Figure 4.2: Relative errors for haplotype abundance estimation versus true strain frequencies. Results were evaluated per method, per data set, and binned by true frequency into bins of size 0.05. Plots show the average relative error per bin. True frequencies were normalized per assembly, taking only the assembled sequences into account for a fair comparison. Only assemblies containing at least 2 strains were evaluated. Plots for the simulated HIV data at 500x and 1000x are similar to HIV-5000x and presented in the Supplementary Material.

for increased error rates on the HIV pol data is the complexity of the data set, with very low frequency strains in combination with a low total coverage (500x–5000x). A more detailed analysis can be found in the Supplementary Material.

Figure 4.2 shows the true haplotype frequencies versus the relative error[1] per method. Results are clustered by true abundance into bins of size 0.05 and average errors are plotted for each bin. On the Poliovirus data there are no results for ShoRAH or PredictHaplo, because the first could not process this data set while the latter found less than a single strain. aBayesQR could only process the HIV pol data; on this data set, Virus-VG, PredictHaplo, and aBayesQR have a similar error pattern, with values well below the errors made by ShoRAH. On HCV and ZIKV data, however, we observe that Virus-VG outperforms the other methods in terms of frequency estimation, with estimates that are closest to the true values. An immediate interpretation of these findings is that accuracy in estimating abundance is inevitably linked with accuracy in haplotype reconstruction, which may explain our overall advantages.

4.3.7 Runtime evaluation

By their worst-case runtime complexity, both candidate path generation and minimizing for selecting optimal sets of haplotypes reflect exponential procedures in Virus-VG. In practice, however, this is not an issue: on our benchmarks Virus-VG is 2.5–87 times

[1] $|x - x^*| / (0.5(x + x^*))$

faster than SAVAGE, which together form our de novo assembly pipeline. Combined, this pipeline takes 43–286 CPU hours on full-length data sets of 20,000x coverage. This is slower than PredictHaplo (2.0–7.4 CPU hours) but faster than ShoRAH (351–814 CPU hours), both of which are reference-guided. However, SAVAGE and Virus-VG can use multiple cores while PredictHaplo does not, leading to comparable wall clock times when multithreading. We present a more detailed runtime and memory analysis in the Supplementary Material.

4.3.8 Summary

We have benchmarked three de novo assembly tools (SAVAGE, Virus-VG, and PEHaplo) and three reference-guided methods (aBayesQR, PredictHaplo, and ShoRAH). The only methods that are stable with respect to all data sets considered, both full-length genomes and shorter regions, are SAVAGE and Virus-VG. Although SAVAGE achieves lowest error rates, Virus-VG is able to build full-length haplotypes with error rates slightly higher than SAVAGE, but still much lower than other methods. PredictHaplo performs well on the labmix at 20.000x, full-length genome and pol region, but it misses many haplotypes on all other data sets, with only 14–64% of the target genomes reconstructed. Virus-VG yields most accurate frequency estimates on full-length genomes, and performs similar to aBayesQR on the HIV pol region. In terms of CPU time, the combination of SAVAGE and Virus-VG is on a par with or faster than all other methods except PredictHaplo, which is consistently faster; in terms of wall clock time, however, both SAVAGE and Virus-VG achieve a major speedup by using multithreading, while PredictHaplo does not.

4.4 Discussion

We have presented an algorithm that turns viral strain-specific contigs, such as available from a de novo assembler like SAVAGE [7], into full-length, viral strain-specific haplotypes, without the use of a reference genome at any point. We first construct a contig-variation graph, which arranges haplotype-specific contigs sampled from a viral quasispecies in a convenient and favorable manner. We then enumerate all maximal-length paths through this graph that maximally concatenate the contig subpaths. Last, we solve a minimization problem that assigns abundance estimates to maximal-length paths that are optimal in terms of being compatible with abundances computed for the nodes in the graph. We finally output the optimal such set of paths together with their abundances, by which *we have completed the de novo viral quasispecies assembly task*.

In benchmark experiments, we have demonstrated that our method yields major improvements over the input contigs in terms of assembly length, while preserving

high accuracy in terms of error rates. Compared to state-of-the-art viral quasispecies assemblers—all of which operate in a reference genome dependent manner—our method produces haplotype-resolved assemblies that are both more complete, in terms of haplotypes covered, and more accurate, in terms of error rates. We believe that (a) this reflects the strength of a fully *de novo* approach, because we avoid to deal with reference-induced biases. We also believe that (b) this is a result of directly integrating haplotype abundance estimation into reconstruction of haplotypes.

Still, improvements are possible. Our current optimization problem employs the absolute difference to determine the abundance estimation error. As future work, we consider the exploration of probabilistic error models, e.g., by modelling path abundance as being Poisson distributed [79] and calculating the likelihood of the observed node abundances.

Further, we had already alluded to that the number of candidate paths is exponential in the number of input contigs, which could theoretically be overwhelming when dealing with highly fragmented assembly output. Our runtime benchmarks show that this is not an issue with standard data sets. Nevertheless, we will consider more efficient alternative solutions in future work, based on a flow formulation of the problem that we recently found, yielding a yet to be implemented polynomial time algorithm.

CHAPTER 5

VIRAL QUASISPECIES RECONSTRUCTION VIA CONTIG ABUNDANCE ESTIMATION IN VARIATION GRAPHS

In this chapter we re-examine the combinatorial properties of the variation graphs constructed in the previous chapter. While we have presented a complete solution to the de novo viral quasispecies assembly problem, there are still some practical problems to tackle in order to deal with data sets of high complexity (such as large genomes or a great number of closely related haplotypes).

Here, we overcome these hurdles by reformulating the combinatorial problem. We solve the contig abundance estimation problem and propose a greedy algorithm to efficiently build full-length haplotypes. Finally, we obtain accurate frequency estimates for the reconstructed haplotypes through linear programming techniques. Together with the work in Chapters 2 and 4, this chapter presents the first de novo approach to successfully and efficiently reconstruct viral quasispecies at full length.

Based on:

J.A. Baaijens, L. Stougie, A. Schönhuth. Strain-aware assembly of genomes from mixed samples using variation graphs. *bioRxiv* 645721, 2019 (submitted).

Supplementary material: https://doi.org/10.1101/645721.

5.1 Background

In comparison to bacteria and eukaryotes, viruses have relatively short genomes that are subject to very high mutation rates; RNA viruses even more so than DNA viruses [35]. As a consequence, the virus particles within most RNA virus infections do not share a single genomic sequence, but rather exist as a cloud of closely related mutant strains: a *viral quasispecies* [32]. These mutant clouds enable viruses to adapt to their environment and possibly escape medical treatment or the host immune response [28, 125]. The mutant strains may show different phenotypic properties and appear at varying frequencies within the population. Viral quasispecies assembly aims to reconstruct each of these individual genomes (or *haplotypes*) and to estimate the corresponding relative abundances.

Besides viral quasispecies, many other genomic data sets contain mixtures of closely related sequences, such as bacterial mixtures or environmental samples (metagenomics). Again, strains may appear at varying frequencies, with some highly abundant and others very rare. Reconstructing all of the individual haplotypes present in such a scenario, more generally known as *haplotype-aware genome assembly*, is a major challenge and requires specialized tools [107]. Here, we offer a solution to the viral quasispecies assembly problem that has the potential to scale to bacterial-size genomes.

We present VG-flow, a method for haplotype reconstruction with integrated abundance estimation. Because genomes from mixed samples are usually affected by substantial mutation rates, VG-flow avoids using standard linear reference genomes altogether. Instead, VG-flow is based on variation graphs as underlying, flexible-to-construct reference systems that account for haplotype-specific mutations in an unbiased manner. We construct these variation graphs in a de novo manner, that is, without resorting to any external means such as a reference genome.

VG-flow takes as input a next-generation sequencing (NGS) data set and a collection of strain-specific contigs assembled from the data, and produces full-length haplotypes and corresponding abundance estimates. Our method is centered on estimating contig abundances in a *contig-variation graph*, a graph that captures all quasispecies diversity present in the contigs [43, 90]. We build a *flow network* to accompany the variation graph and estimate contig abundances by solving a flow-like optimization problem: variables represent flow values on the edges of the flow network and we impose flow constraints, while the objective function evaluates the difference between estimated contig abundances and read coverage for every node in the variation graph. This objective function is convex, hence the flow problem is polynomial time solvable [86].

The flow solution presents abundance estimates for the input contigs, which are of value in its own right in various mixed sample applications [18, 41, 68]. We use the contig abundance estimates in a combination of greedy algorithms to extract candidate

haplotypes from the variation graph, where candidate haplotypes reflect concatenations of subpaths associated with the input contigs. Finally, we solve an optimization problem whose variables represent the haplotype abundances and the difference between read coverage and haplotype abundance is minimized over all nodes. Thus, we obtain a selection of candidate haplotypes that represents the quasispecies, along with haplotype abundance estimates.

Existing viral quasispecies assemblers include widely evaluated tools like [97, 98, 130], as well a variety of methods introduced more recently [3, 7, 8, 10, 22, 57]. These methods can be divided into two classes: reference-guided and reference-free (also referred to as *de novo*). De novo approaches do not require any prior information, such as a reference genome or knowledge of the quasispecies composition. This has been shown to have advantages over reference-guided reconstruction, since using a reference genome can induce significant biases. Especially at the time of a viral disease outbreak, an appropriate reference genome may not be available due to high mutation rates. However, most of the aforelisted tools are reference-guided; only [7], [8], and [22] present de novo approaches.

Moreover, many of these specialized viral quasispecies assemblers aim at single gene reconstruction, rather than whole genome assembly. In [8] we took a first step towards full-length de novo viral quasispecies assembly. There, we have shown that de novo haplotype reconstruction with integrated haplotype abundance estimation yields assemblies that are more complete, more accurate, and provide better abundance estimates. While this approach is guaranteed to find a selection of haplotypes that is optimal in terms of being compatible with the read coverages, its runtime is exponential in the number of contigs. Since the number of contigs generally increases on increasing genome length, we found [8] unsuitable for genomes larger than ~ 10 kb (depending on the number of strains). With VG-flow, we provide a reference-free solution to the full-length viral quasispecies reconstruction problem that scales well to longer genomes. As another benefit of the theoretical rigorosity of our problem formulation and efficiency of the solution, we also experience considerable improvements in terms of accuracy compared to existing tools.

Some of the challenges that have to be dealt with in viral quasispecies assembly can also be found in RNA transcipt assembly, where the goal is to reconstruct an unknown number of transcripts and predict the relative transcript abundances. Not surprisingly, many RNA transcipt assemblers define graph optimization problems similar to our flow formulation [13, 93, 101, 117, 120]. Although dealing with related problems, these methods cannot be applied in a viral quasispecies setting so easily: they require a collection of reference genomes representing all possible haplotypes as input, which is not available in our setting. Nevertheless, the theory behind these approaches is related to what we do. In [117], node and edge abundance errors are used to define a min-cost

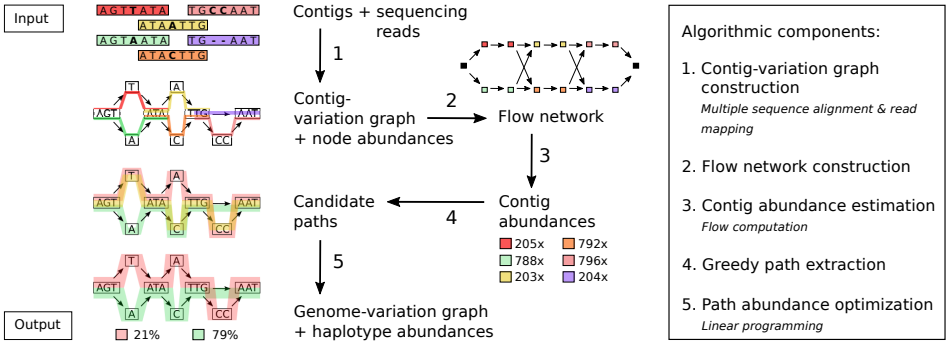

Figure 5.1: Algorithm overview

flow problem; note that this formulation does not take subpath constraints into account. On the other hand, [101] describes how subpath constraints can be incorporated into a minimum path cover formulation. This results in an optimization problem that is solvable in polynomial time, but does not minimize node abundance errors. We use the best of both worlds by defining an optimization problem that takes subpath constraints, minimizes node abundance errors, and is polynomially solvable; this establishes a theoretical novelty. Because this novelty gives way to different types of analyses in other settings, and immediately connects to extensively treated theoretical issues [101, 117], we feel that it is of value also in its own right.

Among more generic assemblers, SPAdes [9] has been shown to be capable of reconstructing individual haplotypes from mixed samples, up to a certain degree. This method was designed for bacterial genomes and scales well to human genomes, but is unable to reconstruct low-frequent haplotypes [7]. Haplotype-aware assembly of metagenomes is a big challenge, which tends to result in scattered genome fragments and missing strains [107]. Metagenomic assemblers such as [15, 62, 89, 92] aim to reconstruct mixtures of viral and bacterial populations at strain level. The contigs obtained with these methods, or any other assembler, can also be used as input for VG-flow. Although we focus on viral quasispecies reconstruction, the mathematical framework presented here is generic and could be applied in other scenarios as well; as such, VG-flow has the potential to make a big step ahead in haplotype-aware genome assembly in general.

5.2 Results

We present VG-flow, a new approach to haplotype-aware genome assembly from mixed samples. This algorithm takes as input a data set of next-generation sequencing reads

and a collection of strain-specific contigs; note that de novo assembly into strain-specific contigs can be performed using various tools (e.g. [7, 9, 22, 89], depending on the application). The output of VG-flow consists of maximal length haplotypes along with relative abundance estimates for each of these sequences. In addition, the algorithm yields abundance estimates for each of the input contigs.

Our approach consists of five steps, as depicted in Figure 5.1:

(1) We construct a *contig-variation graph* VG_C by performing Multiple Sequence Alignment (MSA) on the input sequences. Node abundances are obtained by mapping sequencing reads to the variation graph.

(2) We build a *flow network* FG using VG_C.

(3) We define and solve a flow-like optimization problem on FG to obtain contig abundance estimates.

(4) We generate a set of candidate haplotypes P_{cand} based on the estimated contig abundances through multiple greedy heuristics.

(5) We obtain a selection of haplotypes H from P_{cand} by solving another linear optimization problem, defined on VG_C. The solution to this problem presents estimates for the relative abundances of all candidate haplotypes in P_{cand}, thereby eliminating any false haplotypes.

The final output is presented as a *genome-variation graph* VG_H capturing the haplotypes in H, along with the estimated relative abundances.

Steps (1) and (5) are based on the Virus-VG algorithm [8] and are used without further adjustment. Steps (2), (3) and (4) are entirely novel. They incorporate a new problem formulation, and based on the solution of this problem, provide a way to estimate contig abundance, as part of an overall efficient alternative to the exponential brute-force routines from [8]. VG-flow easily scales to data sets of higher complexity and thus provides a clear view towards haplotype reconstruction for mixtures of bacterial strains or even metagenomic data. Further details on algorithm design are presented in Section 5.5.

5.2.1 Benchmarking preliminaries

We perform benchmark experiments where we compare VG-flow to existing methods for full-length viral quasispecies reconstruction. In these experiments, we make use of the specialized de novo viral quasipecies assembler SAVAGE [7] for generating a set of strain-specific contigs. We compare performance of VG-flow to Virus-VG [8], another de novo approach, and to reference-guided viral quasispecies reconstruction tools PredictHaplo [97] and ShoRAH [130]. More recent viral quasispecies assemblers

Data set	Data type	Virus type	Genome size (bp)	Strain count	Strain abundance	Pairwise divergence
HCV mix	Simulated	HCV-1a	9273–9311 bp	10	5–19 %	6–9 %
ZIKV mix	Simulated	ZIKV	10251–10269 bp	15	2–13 %	1–10 %
Poliovirus mix	Simulated	Poliovirus	7428–7460 bp	6	1.6–51 %	1.2–7 %
Labmix	Real	HIV-1	9478–9719 bp	5	10–30 %	1–6 %

Table 5.1: Quasispecies characteristics of benchmark data sets. All data sets consist of Illumina Miseq reads with an average sequencing depth of 20.000x.

aBayesQR [3], QSdpR [10], and PEHaplo [22] focus on reconstruction of relatively short genomic regions and were unable to process full-length quasispecies data sets at ultra-deep coverage. We provide the reference-guided methods with a *consensus reference genome* obtained by running VICUNA [129] on the data set. This procedure simulates a de novo setting where the viral agent and its genome may be unknown. Moreover, the consensus reference sequence may be a more accurate representation of the data set under consideration than the standard reference genomes available.

We evaluate all assemblies by comparing the assembled contigs to the ground truth sequences using QUAST [47]. This assembly evaluation tool aligns the assembled contigs to the true haplotypes, which are provided as a reference, and calculates several standard evaluation metrics. For each assembly, we report the number of contigs, percent target genomes covered, N50, NG50, and error rate. If an assembly consists of only full-length contigs, the number of contigs can be interpreted as the estimated number of strains. Target genome coverage is defined as the percentage of aligned bases in the true haplotypes, where a base is considered aligned if there is at least one contig with at least one alignment to this base. The N50 and NG50 measures reflect assembly contiguity. N50 is defined as the length for which all contigs in the assembly of at least this length together add up to at least half of the total assembly size. NG50 is calculated in a similar fashion, except that the sum of contig lengths is required to cover at least half of the total target length. Error rates are equal to the sum of mismatch rate, indel rate, and N-rate (ambiguous bases). We do not report unaligned bases or misassemblies as we did not encounter any of these.

In addition to the above QUAST assembly metrics, we evaluate strain abundance estimates by comparing estimated values to true strain abundances. For each assembly, let n be the number of true strains and let x_i, x_i' denote the estimated and true abundance, respectively, of strain i. For each ground truth haplotype, the abundance estimates of sequences assigned to this haplotype were summed to obtain the strain abundance estimate x_i. We only evaluate abundances for strains that are present in the assembly (i.e. $x_i > 0$) since we cannot expect an assembler to estimate the abundance of a missing

strain. Therefore, the true strain abundance values x'_i are also normalized, taking only the assembled sequences into account. Then, we calculate the absolute frequency error (AFE) and the relative frequency error (RFE) a follows:

$$\begin{aligned} \text{AFE} &= \sum_{i \in I} \frac{|x_i - x'_i|}{|I|}, \\ \text{RFE} &= \sum_{i \in I} \frac{|x_i - x'_i|}{|I| \cdot x'_i}, \text{ where} \\ I &= \{i \in [n] : x_i > 0\} \end{aligned}$$

5.2.2 VG-flow scales well to bacterial-size genomes

Our main goal of designing VG-flow was to enable generation of high-quality assemblies for data sets of higher complexity compared to what is possible with other de novo approaches. While Virus-VG [8] performs well on the quasispecies benchmark data sets, the path enumeration step will quickly become too expensive as data sets become more complex: the number of candidate haplotypes is exponential in the number of input contigs. In particular, the number of haplotypes grows exponentially in the genome size, making candidate path enumeration infeasible for larger haplotypes.

In order to explore the limits of VG-flow and to highlight its advantages over Virus-VG, we simulated 28 data sets of increasing complexity using SimSeq [112] (2×250 bp paired-end reads, Illumina MiSeq error profile). We created data sets with genomes of increasing size (2500 bp, 5000 bp, 10.000 bp, 20.000 bp, 40.000 bp, 100.000 bp, 200.000 bp) and an increasing number of strains (2, 4, 6, 8). Each data set has a total coverage of 1000x. Each strain was created by randomly introducing mutations at a mutation rate of 0.5% into a randomly generated nucleotide sequence of the desired length; hence, haplotypes have a pairwise divergence of 1%. For data sets of 2, 4, 6, and 8 strains, the relative strain abundances were set to ratios of 1:2, 1:2:3:4, 1:2:3:4:5:6, and 1:2:3:4:5:6:7:8, respectively. Although the genome sequences are artificial (allowing us to vary genome size and number of strains flexibly) the relative abundances and pairwise divergence reflect plausible real-world, and challenging scenarios in metagenomics [107].

In Figure 5.2 we show VG-flow runtimes as a function of the genome size for a fixed number of strains. As expected, runtime increases as the genome size and the number of strains increase. Even the data sets with a genome size of 200.000 bp are easy to process with VG-flow. Virus-VG, on the other hand, was unable to process any genomes larger than 20.000 bp (2 strains), 5000 bp (4 strains), or 2500 bp (>4 strains).

Remark. Currently, the limiting factor for processing genomes larger than 200.000 bp with VG-flow is the pre-assembly step. VG-flow requires pre-assembled strain-specific contigs as input and we use SAVAGE [7] for this. SAVAGE has proven to produce assemblies of very high quality, but this assembler does not scale well to large genomes.

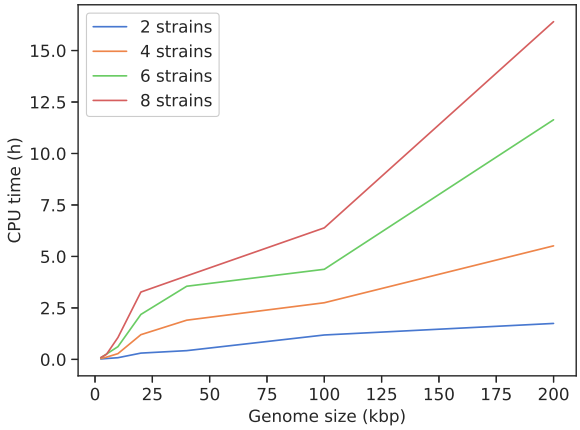

Figure 5.2: VG-flow runtime (CPU seconds) on data sets of increasing genome size (2500, 5000, 10.000, 20.000, 40.000, 100.000, 200.000) and number of strains (2, 4, 6, 8). Note that these runtimes do not include SAVAGE assembly time; a comparison of total runtime is shown in the Supplementary Material.

Inspired by results from [7], we experimented with SPAdes [9] assemblies as input for VG-flow. Although SPAdes does not produce strain-specific contigs as accurately as SAVAGE, it performs reasonably well and VG-flow is able to build full-length haplotypes from these contigs. Results and further details are shown in the Supplementary Material.

5.2.3 VG-flow outperforms existing tools

We evaluate performance of VG-flow on three simulated viral quasispecies data sets from [8] and one real HIV benchmark presented in [30], also referred to as the *labmix*. The simulated data sets are based on true genomic sequences from the NCBI nucleotide database; the characteristics of all data sets (virus type, genome size, number of strains, relative strain abundances, and pairwise divergence) are described in Table 5.1. All data sets consist of Illumina Miseq reads with an average sequencing depth of 20.000x. For each data set, including the labmix, the true haplotypes and their relative abundances are known.

Table 5.2 presents assembly statistics for all methods on the three simulated data sets (HCV, ZIKV, and Poliovirus) and Table 5.3 presents results on the labmix. The de novo approaches VG-flow and Virus-VG both use the contigs obtained with SAVAGE as input. We observe that both methods produce full-length haplotypes for all simulated

	# contigs*	target (%)	N50	NG50	ER(%)	AFE(%)	RFE(%)
10-strain HCV mix							
SAVAGE	26	99.4	8964	8964	0.001	-	-
Virus-VG	10	99.3	9281	9203	0.001	0.1	0.9
VG-flow	10	99.3	9281	9203	0.001	0.0	0.2
PredictHaplo	9	73.8	7636	7608	0.059	0.9	11.3
ShoRAH	639	56.9	7570	7570	4.294	8.5	64
15-strain ZIKV mix							
SAVAGE	100	98.8	2954	3801	0.023	-	-
Virus-VG	20	92.8	10202	10210	0.115	0.3	6.0
VG-flow	21	92.8	10193	10210	0.108	0.3	5.4
PredictHaplo	8	53.3	10270	10267	0.126	4.9	69
ShoRAH	493	26.3	10117	10117	4.392	39	229
6-strain Poliovirus mix							
SAVAGE	59	83.7	1089	1643	0.019	-	-
Virus-VG	14	80.7	7316	7428	0.064	0.6	12.8
VG-flow	12	90.2	7316	7428	0.036	0.3	3.5
PredictHaplo	3	16.6	7461	-	1.825	-	-

Table 5.2: Assembly results on simulated data (Illumina MiSeq, 20.000x coverage). ER = Error Rate (N's + mismatches + indels), AFE = Absolute Frequency Error, RFE = Relative Frequency Error. Frequency errors were only computed for assemblies containing at least 2 full-length haplotypes. *If contigs are full-length, this number reflects the estimated number of strains in the quasispecies.

data sets, with much higher N50 and NG50 values than SAVAGE. The improved assembly contiguity comes with only slightly higher error rates compared to the SAVAGE contigs. Table 5.2 shows that VG-flow builds contigs with even lower error rates than Virus-VG (0.108% versus 0.115% on ZIKV data and 0.036% versus 0.064% on Poliovirus data for VG-flow and Virus-VG, respectively). On the Poliovirus data set we do not only observe a lower error rate for VG-flow compared to Virus-VG, but also a higher target coverage (90.2% for VG-flow versus 80.7% for Virus-VG). The frequency estimation errors (AFE and RFE) in Table 5.2 show that the increase in assembly accuracy also leads to lower frequency estimation errors.

On real data (Table 5.3) we observe that VG-flow produces the same number of contigs as Virus-VG, leading to identical target coverage and N50 values; the only difference between the assemblies is a slightly lower NG50 for VG-flow (4608 versus 4642) and a slightly higher error rate (0.535% versus 0.324%). These differences may be explained by the highly uneven coverage of this data set, which affects the contig abundance

	# contigs	target (%)	N50	NG50	ER(%)
SAVAGE	68	97.9	1026	1450	0.066
Virus-VG	23	90.6	2130	4642	0.324
VG-flow	23	90.6	2130	4608	0.535
PredictHaplo	6	100.0	8825	8825	1.066
ShoRAH	250	100.0	8775	8775	3.910

Table 5.3: Assembly results on the labmix (5-strain HIV mixture, real Illumina MiSeq, 20.000x coverage).

estimation and hence also the greedy path extraction (see Figure 5.1). However, the contigs produced by VG-flow are much longer than the input contigs, with the N50 value more than doubled.

Compared to the state-of-the-art for full-length viral quasispecies reconstruction, we notice a clear advantage for VG-flow in terms of target coverage and error rate. Table 5.2 shows that PredictHaplo and ShoRAH are unable to reconstruct all haplotypes in any of the simulated data sets. The strains that could be reconstructed have higher error rates than VG-flow, as well as much higher frequency estimation errors. On the labmix, PredictHaplo and ShoRAH both achieve a target coverage of 100%. In other words, they assemble each of the five HIV strains at full length. However, PredictHaplo does so at almost twice the error rate of VG-flow (1.066% for PredictHaplo versus 0.535% for VG-flow) and the ShoRAH assembly has an even higher error rate of 3.910%. Moreover, ShoRAH greatly overestimates the number of strains in all data sets considered.

Figure 5.3 shows the relative frequency estimation errors per method as a function of true abundance per strain, for each of the simulated data sets. Results are divided into bins (binsize=0.05) and average errors are shown. Figure 5.3 highlights the advantage of de novo methods VG-flow and Virus-VG, which have much smaller relative errors than PredictHaplo and ShoRAH. On the HCV and ZIKV data sets, VG-flow and Virus-VG show nearly identical performance; on the Poliovirus data, we observe a small advantage for VG-flow.

5.2.4 De novo approaches achieve highest precision and recall

In addition to the standard assembly quality metrics presented in the previous section, we analyze relevance of the reported solutions and the amount of similarity between true and reconstructed haplotypes. In the following, we define true positives by their relative edit distance to the corresponding true haplotype (i.e., edit distance divided by alignment length). A contig is considered a true positive if it aligns to a true haplotype with relative edit distance $\leq \alpha$; a haplotype is considered correctly reconstructed if at

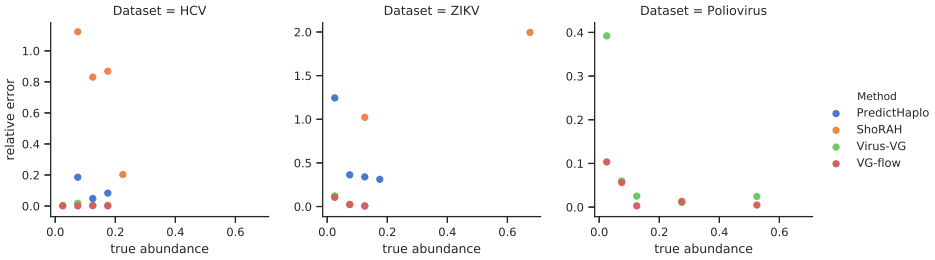

Figure 5.3: Abundance estimation results per data set. Abundances were only evaluated for assemblies containing at least 2 full-length haplotypes.

least one contig aligns to it with relative edit distance $\leq \alpha$. Figure 5.4 presents precision, recall, and F-measure per data set. These measures evaluate the number of true positive contigs relative to the total number of contigs (precision), the number of correctly reconstructed haplotypes relative to the total number of haplotypes (recall), and the harmonic average of precision and recall (F-measure = 2*precision*recall/(precision+recall)). We consider various thresholds for the relative edit distance and plot precision, recall, and F-measure as a function of the threshold α. Each of these measures takes values between 0 and 1, with 0 the worst possible score and 1 the best possible score.

We observe that, in general, SAVAGE achieves high values for all three measures already at low relative edit distance. However, SAVAGE only assembles short contigs. VG-flow and Virus-VG show very similar performance, with slightly better values for VG-flow on the ZIKV and Polio data sets, and slightly better performance of Virus-VG on the labmix. All other methods are outperformed by these de novo approaches: PredictHaplo and ShoRAH do not achieve comparable F-measure scores on the simulated data. On the labmix these methods obtain similar scores only at an allowed relative edit distance of 4%, which is nearly as high as the maximal pairwise divergence between strains in this data set.

5.2.5 Runtime and memory usage

The haplotype reconstruction steps used in VG-flow are highly efficient: on the benchmark data sets presented in Table 5.1 we measured a decrease in haplotype reconstruction time of 9.2–92% compared to Virus-VG. However, total runtime for VG-flow is mostly determined by the contig-variation graph construction step, which involves multiple sequence alignment and read mapping. This graph construction step is shared by VG-flow and Virus-VG. Hence, when considering the complete approach on the simulated quasispecies benchmarks, we observe identical runtime and memory us-

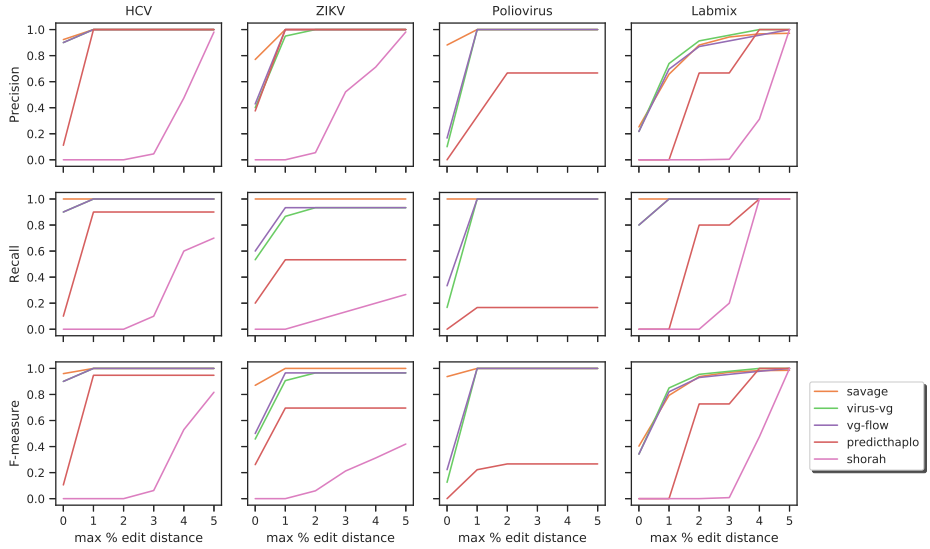

Figure 5.4: Precision, recall, and F-measure per data set.

age for VG-flow and Virus-VG: runtime varies between 3.6–12.5 CPU hours and peak memory usage is between 0.6–0.9 GB. Both approaches require as input pre-assembled contigs, for which we used SAVAGE. This de novo assembler constructs an overlap graph from the sequencing reads, which is an expensive procedure (30.6–276 CPU hours). In comparison, PredictHaplo is faster (2.0–7.4 CPU hours) and ShoRAH is slower (209–814 CPU hours, unable to process the Poliovirus data set). However, it is important to realize that all methods except PredictHaplo are able to profit from multithreading, leading to competitive wall clock times on sufficiently large computing clusters. Moreover, constructing a consensus reference genome to be used for reference-guided methods ShoRAH and PredictHaplo also incurs some additional costs (0.07–0.44 CPU hours). More details are presented in the Supplementary Material.

5.2.6 Analysis of an HCV patient sample

In order to demonstrate utility of VG-flow on real data, we ran our method on a patient sample (plasma) of a Hepatitis C virus infection (subtype 1a). This sample is part of a deep sequencing initiative of HCV genomes [48]. It consists of 349268 reads (2×250 bp, Illumina MiSeq), covering the HCV reference genome (NC_004102.1) from position 2296 to 7328 with an average sequencing depth of 34704x. We performed de novo assembly with SAVAGE and obtained 133 contigs varying in length from 152 to 1238 bp, with an

N50 value of 472. After running VG-flow on this set of contigs, we obtained 33 contigs with an N50 value of 2342. Among the contigs were 7 full-length haplotypes (>4500 bp), in agreement with the analysis performed by [48] using single genome amplification. The estimated relative frequencies varied between 6.0% and 33.3%. Two of the full-length haplotypes show a large insertion of 573 bp at position 3546 of the HCV reference genome; this insertion falls into the NS3 gene, which is involved in viral RNA replication through helicase activity. Overall, the 7 full-length haplotypes have 98.8–99.4% pairwise sequence identity, while they share only 93.8–94.7% of their sequences with the HCV reference genome. The overall assembly process took 92 minutes using 12 CPUs (51 minutes for SAVAGE, 41 minutes for VG-flow) and used 1.2 GB of RAM.

5.3 Discussion

Many genomic data sets contain mixtures of closely related sequences, such as viral quasispecies or bacterial populations, where the number of haplotypes is generally unknown and relative abundances may differ per haplotype. VG-flow addresses these challenges: we successfully reconstructed haplotypes from several mixed samples, both simulated and real, and obtained highly accurate frequency estimates for each haplotype.

VG-flow performs full-length haplotype-aware genome assembly, without using existing linear reference genomes, but by constructing variation graphs from pre-assembled contigs. This approach establishes a reference system that allows for analyses that do not suffer from any kind of haplotype-specific mutation-induced biases. We compute abundance estimates for the input contigs, which are of value in its own right. We also enable haplotype reconstruction in polynomial time, with runtimes depending linearly on genome size in practice. We have shown that VG-flow scales well to bacterial-size genomes, hence proving its potential to contribute also to metagenomic assembly. In benchmark experiments on simulated viral samples, our method outperformed the state-of-the-art in full-length viral quasispecies reconstruction in terms of assembly completeness, assembly accuracy, and abundance estimation quality. Finally, we also demonstrated the value of our method on real HIV and HCV data sets.

In view of the general benefits of Virus-VG [8] and the fact that its brute-force solution experiences severe limitations with respect to genome size and the number of contigs, our main goal was to find a polynomial time solution producing high quality assemblies like Virus-VG. In addition to decisive speed-ups, our algorithm also improved on Virus-VG in terms of assembly accuracy. Virus-VG and VG-flow both aim to reconstruct individual haplotypes from pre-assembled contigs and perform abundance estimation, but VG-flow uses a radically different approach to generating candidate haplotypes. Our results show that the greedy path extraction step in our algorithm

selects a subset of possible haplotypes that represents the quasispecies sufficiently well. By limiting the path abundance optimization to this subset of haplotypes, many false haplotypes are excluded from optimization and hence the algorithm gets less confused by false haplotypes.

Interestingly, for some data sets VG-flow was able to improve on the input contigs in terms of target coverage. A possible explanation is that for haplotypes which are not fully represented by pre-assembled contigs, VG-flow is able to reuse contigs from other strains in the same region. This hypothesis is supported by the fact that error rates for VG-flow are higher than for the input contigs constructed using SAVAGE.

The results in Figure 5.2 show that VG-flow has the potential to process larger genomes. Currently there are two limiting factors that prevent us from processing bacterial or metagenomic data. First, our method requires as input a collection of pre-assembled, strain-specific contigs. We obtained best results using SAVAGE [7] to generate these input contigs; however, in its current state SAVAGE does not scale well to larger genomes. Experiments using SPAdes to generate input contigs have shown that VG-flow can also generate full-length haplotypes from these assemblies. In fact, any haplotype-aware assembler could be used to generate the input contigs, but the quality of the input contigs has a significant impact on the quality of the output.

Another limiting factor is that VG-flow depends on multiple sequence alignment for constructing the contig-variation graph. This step can become quite expensive as the number of contigs grows, which could lead to difficulties when processing metagenomic data sets. Nevertheless, we have moved to a much wider range of feasible genome sizes than what was possible before. Note that variation graph construction is part of a current, very active area of research, such that improvements on that end are to be expected [43, 74, 90].

Note finally that each of these factors imposes the same limitations to the approach in [8]. Addressing these challenges would make VG-flow truly capable of processing bacterial or metagenomic data. Therefore, each of the points discussed above provides an interesting starting point for future work.

5.4 Conclusions

While multiple approaches to reference-free viral quasispecies assembly have been introduced recently, efficient reconstruction of full-length haplotypes without using a reference genome is a major challenge. Although de novo methods have shown advantages over reference-guided tools, the resulting assemblies often consist of rather short contigs. In this chapter we have proposed VG-flow as an efficient solution to extend pre-assembled contigs into full-length haplotypes, based on variation graphs as reference systems that allow for a bias-free consideration of all haplotypes involved.

Benchmark experiments have shown that VG-flow outperforms the state-of-the-art in viral quasispecies reconstruction in terms of accuracy of haplotype sequences as well as abundance estimates. Moreover, we have shown that our method scales well to bacterial-size genomes, thus proving its potential for processing larger data sets like bacterial mixtures or metagenomic data.

5.5 Methods

5.5.1 Variation graphs

Variation graphs are mathematical structures that capture genetic variation between haplotypes in a population [43, 90]. These graphs provide a compact representation of a collection of input sequences by collapsing all shared subsequences between haplotypes.

Definition. Let S be a collection of sequences. We define the variation graph VG_S as a tuple (V, E, P, a). The nodes $v \in V$ store sequences $\text{seq}(v)$ of nucleotides (of arbitrary length) which appear as a substring of some $s \in S$. The edges $(v_1, v_2) \in E$ indicate that the concatenation $\text{seq}(v_1)\text{seq}(v_2)$ also appears as a substring of some $s \in S$. In addition to nodes V and edges E, a variation graph stores a set of paths P representing the input sequences: for every $s \in S$ there is a path $p \in P$ (i.e. a list of nodes, linked by edges) such that the concatenation of node sequences equals s. Finally, we store path abundances using an abundance function $a : P \to \mathbb{R}$ which assigns an absolute abundance value to each path in P.

Approach. Following [8], we distinguish between two types of variation graphs: *contig-variation graphs* and *genome-variation graphs*. Let C be a set of pre-assembled contigs and let H be the collection of haplotypes we aim to reconstruct. The contig-variation graph $VG_C = (V_C, E_C, P_C, a_C)$ organizes the genetic variation that is present in the input contigs and the abundance function a_C gives contig abundance values for every input contig. The genome-variation graph $VG_H = (V_H, E_H, P_H, a_H)$ stores the haplotypes within a population and the abundance function computes haplotype abundances. Constructing a genome-variation graph is the goal of our method; the key idea is to use the contig-variation graph to get there.

5.5.2 Contig-variation graph construction

We construct a contig-variation graph from pre-assembled contigs C using existing techniques for variation graph construction, similar to [8]. This entails three steps:

(1) Multiple sequence alignment (MSA). We run `vg msga` [43] on the input contigs; the resulting MSA is represented as a graph (V, E, P).

(2) Compactification. We compactify the graph by contracting any non-branching path into a single node. For every contig, we update the corresponding path $p \in P$ such that it stores the path through the compacted variation graph. Thus, we obtain a graph (V_C, E_C, P_C).

(3) Node abundance computation. We use `vg map` [43] to align the sequencing reads to the compacted variation graph. From these alignments we compute the average base coverage for every node in the graph, also referred to as the *node abundance*.

Note that the computed node abundances do not yet give us the abundance function $a_C : P_C \to \mathbb{R}$. To complete the construction of VG_C, we construct a flow network and solve a minimum-cost flow problem as described below.

5.5.3 Flow network construction

We construct a flow network $FG = (V, E, c, d)$, which allows us to compute contig abundances by solving a variant of the minimum-cost flow problem. Network flows are defined on directed graphs, where every edge has a given capacity and receives a certain amount of flow [4]. A flow network has a *source* node and a *sink* node, which have only incoming and outgoing flow, respectively. For all other nodes, the amount of incoming flow must always equal the amount of outgoing flow, so-called flow conservation. We define our graph as follows.

Nodes: we start by creating a source s and a sink t. Then, we introduce two vertices for every contig $c_i \in C$, thus obtaining the vertex set $V = \{s, t\} \cup \{v_i^-, v_i^+ \mid c_i \in C\}$.

Edges: we introduce directed edges (arcs) of three types: *contig-arcs*, *overlap-arcs*, and *auxiliary-arcs*. For each contig c_i we add a contig-arc $e_i : v_i^- \to v_i^+$. For each pair of contigs c_i and c_j there is an overlap-arc e_{ij} from vertex v_i^+ to vertex v_j^- if a suffix of c_i has a non-conflicting overlap with a prefix of c_j. In other words, the sequences of c_i and c_j are identical on their overlap. Finally, we add auxiliary-arcs $s \to v_i^-$ for any v_i^- which has no incoming overlap-arcs, and auxiliary-arcs $v_i^+ \to t$ for any v_i^+ which has no outgoing overlap-arcs.

Capacities: all edges have infinite capacity.

Costs: to every edge $e \in E$, we assign a cost d_e where

$$d_e = \begin{cases} 1, & \text{for contig-arcs;} \\ -1, & \text{for overlap-arcs;} \\ 0, & \text{for auxiliary-arcs.} \end{cases}$$

The intuition behind this construction is that haplotypes can be found as $s-t$ paths in FG and flow along the edges reflects accumulated haplotype abundances. The edge

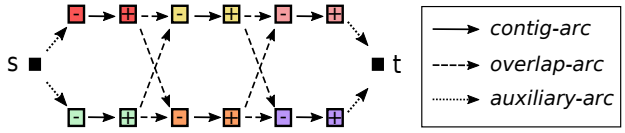

Figure 5.5: Flow network construction: source (s), sink (t), vertices (v_i^-, v_i^+), contig-arcs, overlap-arcs, and auxiliary-arcs.

costs in the flow network allow for the definition of a minimum-cost flow problem that computes contig abundances that are optimal in terms of being compatible with the node abundances in the contig-variation graph, as described in the next section. The construction of the flow network is illustrated in Figure 5.5.

5.5.4 Contig abundance computation

The problem of estimating contig abundances has applications in metagenomics [64] and RNA transcript assembly [26]. Existing methods make use of read mapping, either to a reference genome [68] or to the contigs [18, 41]. Such techniques may cause ambiguous alignments when contigs overlap or share identical sequence. Here, we avoid these issues by mapping reads to the contig-variation graph and solving a flow-like optimization problem.

Problem formulation. Candidate haplotypes in the contig-variation graph VG_C can be obtained by concatenating overlapping contig subpaths. Therefore, any maximal-length path in the variation graph corresponds to an s-t path in FG. We denote by $\delta^+(v)$ and $\delta^-(v)$ the set of arcs, respectively, entering and leaving $v \in V$. Recall that V_C denotes the set of nodes in VG_C and let a'_u denote the abundance of node $u \in V_C$, as computed from the read alignments. For a node $u \in V_C$ and edge $e \in E$, we write $u \in e$ if the contig (or overlap) associated with the contig-arc (or overlap-arc) passes through node u in the contig-variation graph. We define the following flow problem, in which the variables x_e decide the amount of flow going through arc $e \in E$:

$$\begin{aligned} \min \quad & \textstyle\sum_{u \in V_C} |a'_u - \sum_{\{e \in E | u \in e\}} d_e x_e| \\ \text{s.t.} \quad & \textstyle\sum_{e \in \delta^+(v)} x_e = \sum_{e \in \delta^-(v)} x_e, \ \forall v \in V \setminus \{s,t\} \\ & x_e \geq 0, \ \forall e \in E. \end{aligned} \tag{5.1}$$

Motivation. The objective function evaluates the node abundance errors, defined as the absolute difference of the node abundance and the sum of contig abundance estimates of all contigs whose path passes through the node under consideration. However, contigs belonging to the same haplotype may have overlaps due to conserved regions between haplotypes; we need to avoid double-counting the contig abundances

for nodes corresponding to such overlaps. The edge costs d_e ensure that for any pair of overlapping contigs, for any node $u \in V_C$ in the overlap, the estimated abundance is only added to the sum once.

Solution. The objective function is convex in the flow-variables x_e (Supplementary Material, Lemma 3.1), so we have the problem of minimizing a convex function over a set of linear constraints. Such problems can be solved in polynomial time [86]. Given a solution to this optimization problem, the flow values on the contig-arcs reflect contig abundance estimates. We use these values to define the abundance function a_C on the contig-variation graph. An evaluation of abundance estimates for all simulated data sets is presented in the Supplementary Material. Below, we explain how to use these contig abundances to extract candidate haplotypes for further optimization.

5.5.5 Greedy path extraction

The outcome of the above algorithm gives us a flow value for each edge in the flow network. In the biological context of the problem, we are interested in a decomposition of this flow into a set of s-t paths representing the reconstructed haplotypes. Finding such a flow decomposition can be done in polynomial time, as follows from any constructive proof of the Flow Decomposition Theorem [4]. In general, we are interested in a parsimonious solution (i.e. a solution with a small number of paths). Finding a decomposition with a minimal number of paths, however, is NP-complete. Many approximation algorithms have been developed for finding a minimum path flow decomposition, e.g. [56, 108], but these algorithms could not even handle our smallest data set (a mixture of 2 haplotypes of length 2500 bp). Therefore, we resort to other, more efficient means for obtaining a set of haplotypes from the given flow solution [122].

We consider a generic greedy heuristic to obtain a selection of candidate paths (Algorithm 1). This approach iteratively selects an $s-t$ path p from the flow network, then updates the flow solution by subtracting the largest possible flow on contig-arcs in p. It terminates when all flow on contig-arcs is below a user-defined threshold for the minimal haplotype abundance. The order in which paths are selected depends on the optimality criterion: we consider maximum capacity paths, minimum capacity paths, and shortest paths—this essentially gives rise to three heuristics. Note that we do not take the flow values on overlap-arcs or auxiliary-arcs into account, because we want to avoid any preliminary restrictions on the contig overlaps used.

It varies per data set which optimality criterion gives best results: the maximal capacity criterion extracts paths in order of decreasing abundance, hence leads to paths which are most reliable. However, if a sample contains low-frequency strains, it can be beneficial to select haplotypes in order of increasing abundance (minimum

Algorithm 1 Greedy path extraction from a given flow solution

Input: flow network FG, contig-arcs E', flow solution x, min abundance m, optimality criterion OC

Output: a selection of candidate paths P_{cand}

function GREEDYPATHS(FG, E', x, m, OC)
  $R \leftarrow x$
  $P_{\text{cand}} \leftarrow \emptyset$
  $FG_R \leftarrow FG$
  while FG_R has at least one $s-t$ path **do**
    Find an $s-t$ path p in FG_R that is optimal w.r.t. OC
    $w \leftarrow \min_{e \in p \cap E'}\{R_e\}$
    $R \leftarrow R - wp$
    $FG_R \leftarrow FG \setminus \{e \in E' : R_e < m\}$
    $P_{\text{cand}} \leftarrow P_{\text{cand}} \cup \{p\}$
  return P_{cand}

capacity). Since we do not know the composition of the quasispecies beforehand, we combine the results of all three heuristics into one set of candidate haplotypes for further optimization. Earlier work has shown that merging a pool of high quality approximations allows for efficient solutions to well-known optimization problems [16, 27]. We compare performance of our combined approach and the individual greedy heuristics in the Supplementary Material.

5.5.6 Path abundance optimization

Given a collection of candidate haplotypes P_{cand} in the form of paths through the contig-variation graph, the only task remaining is to compute relative abundances for these haplotypes. Although the greedy path extraction algorithm produces preliminary path abundance estimates, these can be improved by the following linear programming approach, also described in [8].

Problem formulation. Let a'_v denote the abundance of node $v \in V_C$, which was computed from the read alignments to VG_C. We define variables $x_p \in \mathbb{R}_{\geq 0}$ for $p \in P_{\text{cand}}$, representing the estimated abundance for haplotype p, and consider the following optimization problem:

$$\min \sum_{v \in V_C} \left| a'_v - \sum_{p \ni v} x_p \right| \qquad \text{s.t. } x_p \geq 0 \quad \forall\, p \in P_{\text{cand}}. \tag{5.2}$$

The objective function is similar in spirit to the objective in Equation (5.1), where abundance estimation errors are evaluated per node in the contig-variation graph. Only

now, we compute the absolute difference between the node abundance value and the sum of abundance estimates for all haplotypes passing through this node. This is a convex programming formulation, which can be linearized and solved using an LP solver.

5.5.7 Genome-variation graph construction

Given the candidate haplotypes P_{cand} and the abundance estimates x_p for $p \in P_{\text{cand}}$, we obtain a final selection of haplotypes $H = \{p \in P_{\text{cand}} : x_p \geq m\}$. Here, m is a user-defined minimal path abundance, by default set to 1% of total sequencing depth. Given H, we can transform the contig-variation graph VG_C into the genome-variation graph VG_H, a complete representation of the viral quasispecies.

5.5.8 Data simulation

All synthetic data sets were generated using the software SimSeq [112] to simulate Illumina MiSeq reads from the genome of interest. In order to obtain realistic sequencing error profiles, we used the MiSeq error profile provided with the software. The genomes used for each data set are listed in the Supplementary Material.

5.5.9 Availability of data and material

Software and analysis scripts are publicly available at https://bitbucket.org/jbaaijens/vg-flow. The synthetic benchmark data sets analysed during the current study are available at https://bitbucket.org/jbaaijens/savage-benchmarks. The real HIV data set (labmix) is available at https://github.com/cbg-ethz/5-virus-mix. The real HCV data set is available in the Sequencing Read Archive under accession number SRR3951347.

CHAPTER 6

DISCUSSION

6.1 Overview

The aim of haplotype-aware genome assembly is to reconstruct the copy-specific genomic sequences *(haplotypes)* of an organism from sequencing reads. This has many applications: from analyzing virus infections to patient-donor matching for organ transplantation, and from environmental studies to cancer tumor analysis. Haplotypes may show a lot of variation within a population, leading to problematic reference-induced biases in reference-guided assembly. Reference-free *(de novo)* haplotype reconstruction provides a valuable alternative.

De novo computation of haplotype-specific sequences is a complex task. Sequences within a single sample are often closely related, making it difficult to distinguish between haplotypes and to identify co-occurring mutations. In many application scenarios, the number of haplotypes in a sample is unknown, and relative abundances differ per haplotype. An important component of any algorithm performing haplotype-aware genome assembly is to distinguish sequencing errors from true genomic variation. Beyond this task, a de novo algorithm needs to combine sequencing reads into contiguous sequences *(contigs)* of maximal length; haplotype-specific contigs are also referred to as *haplotigs*. These steps become particularly challenging when haplotypes are represented by reads only at low sequencing depths.

These difficulties have left de novo haplotype reconstruction in a rather immature state for a long time. Recently, significant advances were made using the long reads produced by TGS machines [23, 53, 96, 127]. Be that as it may, the majority of sequencing machines installed worldwide performs short read sequencing, largely dominated by Illumina machines [109]. Enormous quantities of NGS reads have been generated, but these data sets have not yet been fully exploited in terms of haplotype reconstruction. To address this issue, we have presented four methods, each solving a specific component of the haplotype-aware genome assembly problem from NGS reads.

In **Chapter 2** we addressed the viral quasispecies assembly problem, aiming to reconstruct all virus strains present in an infection. Viral quasispecies are polyploid, but in general the exact number of haplotypes *(ploidy)* is unknown. In contrast, the method in **Chapter 3** was designed specifically for polyploid genomes of known ploidy, where data sets are typically of much lower sequencing depths. Both methods make us of the overlap graph assembly paradigm, and succeed in generation of high quality haplotigs that give a complete representation of the individual haplotypes present in the sample.

In **Chapter 4** we extended the strain-specific contigs in viral quasispecies assemblies into full-length haplotypes and assigned relative abundances to each haplotype. In many cases, there are conserved regions between different strains in a quasispecies. If these identical pieces of sequence are longer than the read length, the surrounding variants cannot be assigned to haplotypes by traditional assembly algorithms. Contigs can

therefore not be extended beyond the conserved region and do not grow into full-length haplotypes. We have shown that these ambiguities can be resolved by taking the relative strain abundances into account. By arranging the strain-specific contigs in the form of a variation graph, enumerating candidate haplotypes as maximal-length paths through this graph, and solving a minimization problem that assigns abundance estimates to each of these paths, we were able to complete the viral quasispecies assembly task.

Although **Chapter 4** already enables highly accurate reconstruction of full-length viral genomes, this solution suffers from practical limitations in terms of genome size and number of strains. In **Chapter 5**, we overcame these issues by reformulating the combinatorial problem in such a way that it became polynomially solvable. Through this new solution, we got rid of the practical limitations regarding de novo assembly of viral haplotypes.

6.2 Contributions

The methodology presented in this thesis directly contributes to the reconstruction of viral quasispecies from patient samples. Another immediate application of our methods is in haplotype reconstruction of highly divergent genomic regions, such as the MHC region. In addition to practical contributions, this thesis brings a theoretical contribution in the form of overlap graph construction and variation graph-based optimization. Each of these is described in more detail below.

6.2.1 Viral quasispecies assembly

Determining the individual haplotypes that cause a viral infection can play an important role in therapy selection [32, 39]. In this, another relevant piece of information is the distribution of strain frequencies: which haplotypes are most abundant, and which are very rare? Low-frequency strains are easily suppressed by high-frequency strains within the infection. However, when high-frequency strains are eliminated through a given treatment, the quasispecies composition may change completely. This may have a great impact on the patient's health.

In order to reconstruct low-frequency strains in a viral quasispecies sample, specialized assembly tools are required. As viral genomes are relatively short, one can afford sequencing at ultra-deep coverage (>10.000x), thus (theorically) enabling reconstruction of rare strains. In practice, however, it is very difficult to distinguish sequencing errors from true variation in low-frequency strains. We have made this possible by constructing overlap graphs as the basis of our assembly algorithm in Chapter 2—see also Section 6.2.3. Subsequently, the methods in Chapters 4 and 5 extend these contigs into full-length haplotypes while estimating relative haplotype abundances. Together,

Chapters 2, 4 and 5 form *the first de novo approach to successfully and efficiently reconstruct viral quasispecies at full length.*

We have illustrated the advantages of a de novo approach in several benchmark experiments, considering a variety of virus types, genome sizes, and quasispecies compositions. After evaluating the performance of reference-guided approaches using high-quality reference sequences as well as ad-hoc ("bootstrap") reference genomes, we observed that reference-guided approaches suffer from severe reference-induced biases. Although (generic) de novo assemblers avoid such biases, these methods proved unable to reconstruct low-frequency haplotypes. As a result, our specialized de novo approach outperformed the state-of-the-art in viral quasispecies assembly, as well as in standard de novo assembly.

6.2.2 Haplotype reconstruction in the MHC region

The major histocompatibility complex (MHC) encodes for molecules that are involved in the acquired immune system: the MHC molecules display peptide fragments derived from pathogens on their cell surface, to be recognized and dealt with by immune cells. Therefore, the MHC region plays an important role in infectious, immune-mediated, and autoimmune diseases [24, 77]. Pathogens can evade immune responses if their genomes have mutated such that they are no longer recognized by MHC molecules. However, the MHC is highly polymorphic, meaning that each gene exists in many different variants within the population—in the human genome, there is no region known to be more diverse than the MHC region [54]. The MHC genes are usually divided into two classes; every individual possesses multiple genes for each class, thus representing a wide range of different peptides. While these properties make it difficult for pathogens to escape immune responses, they also render assembly of the MHC region particularly challenging, leaving many secrets still to be discovered.

Similar to our observations in viral quasispecies, the high degree of polymorphism in the MHC asks for a de novo approach to haplotype assembly. In Caskey et al. [19] it has been shown that the methods from Chapter 2 can be applied directly to assemble MHC genes in rhesus macaques. However, the MHC region is three orders of magnitude longer than most RNA virus genomes, and sequencing depths are typically much lower. It is also important to realize that, when the ploidy of a genome is known, one can use this information during the assembly process. In Chapter 3 we presented an algorithm designed specifically for genomes of known ploidy, sequenced at low to medium coverage. Benchmark experiments on simulated data sets containing human MHC haplotypes showed that our method was able to reconstruct these sequences to a high degree of accuracy, thus outperforming all other approaches to haplotype-aware assembly from NGS reads. Since our advancements allow for a more detailed

and accurate analysis, we hope for a deeper understanding of the MHC region through application of our algorithms in medical practice.

An opportunity for future research is the assembly of full-length haplotypes for genomes of fixed ploidy, thus continuing the work in Chapter 3. Although these assemblies were shown to be of very high quality regarding error rates and assembly completeness, there is great potential for algorithms that extend these haplotigs into full-length haplotypes. The approaches presented in Chapters 4 and 5 do not apply to the case of genomes of fixed ploidy: these methods make use of the fact that haplotype abundances vary within a quasispecies, while in genomes of fixed ploidy the haplotypes usually appear at identical frequencies. Hence, adaptation of the methods from Chapters 4 and 5 to the fixed ploidy case provides most interesting future work.

6.2.3 Overlap graph construction

Two commonly used data structures in genome assembly, de Bruijn graphs and overlap graphs, make the foundation of the two most popular paradigms in genome assembly. Nearly all of the NGS based genome assemblers rely on de Bruijn graphs. Thereby, reads are decomposed into k-mers, where k is usually considerably smaller than the read length. In these approaches, sequencing reads are not used to their full potential before the post-processing stages. As mentioned above, it is imperative in haplotype-aware genome assembly to distinguish low-frequency mutations from sequencing errors. While low-frequency mutations are genetically linked, hence co-occur within different reads, sequencing errors do not exhibit patterns of co-occurrence. Examining the full read span decisively enhances the detection of patterns of co-occurrence. For this reason, we opt for overlap graphs as the basis of haplotype-aware genome assembly.

Chapters 2 and 3 present de novo assembly algorithms based on overlap graph construction. These graphs make use of the full read span and do not decompose reads into smaller parts, leading to accurate distinction between sequencing errors and true variants to be assigned to haplotypes. In particular, we calculate an overlap quality score that reflects the probability that a pair of overlapping reads originates from the same haplotype.

The use of efficient indexing techniques has been key to reference-free construction of overlap graphs. We have implemented the most recent version of an algorithm for finding all approximate suffix-prefix overlaps within NGS data [59]. Although this approach is substantially faster and more space efficient than previous algorithms, constructing overlap graphs remains an expensive process. Given the ultra-deep sequencing data sets for viral quasispecies, we have tailored our algorithm in Chapter 2 towards dividing the data into chunks of 500 to 1000x, and merging the contigs of the chunks in subsequent steps. While this works well, it sets certain limits on the frequency

of strains it can recover—haplotypes with frequencies below 1% remain difficult to reconstruct. Further improvements may be achieved by considering alternative indexing techniques or approximation algorithms, thus allowing for faster overlap graph construction.

When processing large genomes in Chapter 3, we make use of a reference genome for binning reads in an initial step; after binning, we discard the reference genome and any related information entirely such that the algorithm operates in full de novo mode. While this binning step does not require a high-quality reference genome, as long as reads get mapped, a truly reference-free approach requires alternative algorithms for computing all-pairs approximate suffix-prefix overlaps.

6.2.4 Variation graph-based optimization

Variation graphs are mathematical objects that can be used as a compact representation of haplotypes within a population, to serve as a reference system that includes all genetic variation. Using such genome structures instead of standard linear reference genomes has been shown to reduce reference-induced bias [31, 90] and to allow for efficient subhaplotype match queries [88] and haplotype modelling [103]. However, variation graph construction has been focused on a linear reference genome as a point of departure. In Chapters 4 and 5, we construct variation graphs from pre-assembled sequences completely *de novo*: first, we sort the contigs in an appropriate way and then we apply progressive multiple alignment techniques [43]. The resulting variation graphs enable full-length haplotype reconstruction through combinatorial optimization, without using any prior information (such as a reference genome).

In Chapter 4, we obtain full-length haplotypes as a selection of maximal-length paths in the variation graph, each of which reflects a concatenation of subpaths associated with the input contigs. The selected paths are optimal in terms of differences between their estimated abundances and the read coverages computed for the nodes they traverse. This approach requires exhaustive path enumeration, where the number of candidate paths is exponential in the number of input contigs; this could theoretically be overwhelming when dealing with highly fragmented assembly output.

Chapter 5 shows how to avoid exhaustive path enumeration by introducing an appropriate flow formulation of the problem. The solution to this optimization problem yields abundance estimates for the input sequences, which are of value in their own right in various mixed sample applications [18, 41, 68]. Note that in both chapters, we minimize differences between estimated haplotype abundances and observed read coverages, while taking the subpaths defined by the input contigs into account.

Although similar optimization problems have been formulated in previous work on viral quasispecies assembly [6, 111], these methods make use of read graphs rather than

variation graphs and define optimality for path abundance estimation in a very different way. Also in RNA transcript assembly, similar problem formulations exist [13, 40, 67, 81, 93, 101, 117, 120]. Most importantly, [101] introduce node and edge abundance errors and [117] show a minimum path cover with subpath constraints to be polynomially solvable. However, none of these approaches simultaneously employs both subpath constraints *and* abundance error minimization in its problem formulation. Hence, the theoretical work in Chapters 4 and 5 establishes a theoretical novelty which may also contribute to the field of RNA transcript assembly—see also Section 6.3.2.

6.3 Future applications

In addition to the immediate contributions described above, there are several future applications and possible extensions of the work presented in this thesis.

6.3.1 Metagenome assembly at strain resolution

Environmental samples usually contain a diverse collection of genomes, also referred to as a metagenome. These sequences come from many different organisms, most of which are of viral or bacterial origin. The complexity and diversity of microbial communities, as well as low divergence between related strains, make assembly of metagenomes a difficult task. In particular, strain-aware assembly is one of the current major challenges in the field of metagenomics.

As a consequence of the efficiency of the algorithms presented in Chapter 5, these techniques have the potential to process larger genomes. By scaling from viral-size genomes to bacterial-size genomes, we take a big step towards strain-aware assembly of metagenomes. However, metagenome assembly brings along a new range of challenges: not only are the genomes larger, they also have a more complicated structure showing repeats and other structural variation. This requires specialized strain-aware de novo assemblers for constructing strain-specific contigs from metagenomic data, and modification of the variation graph construction process. Nonetheless, the methods in Chapter 5 provide an exciting starting point for work on metagenome assembly.

6.3.2 Transcriptome assembly

Some of the challenges that have to be dealt with in viral quasispecies assembly can also be found in RNA transcipt assembly, where the goal is to reconstruct an unknown number of RNA transcripts and predict the relative transcript abundances. Not surprisingly, many RNA transcipt assemblers work in a similar way to the approaches described in Chapters 4 and 5 [13, 93, 101, 117, 120]. As described in Section 6.2.4, some of these methods evaluate node abundance errors in a min-cost flow problem

similar to Chapter 5, while others describe how to incorporate subpath constraints into a minimum path cover formulation. However, none of these methods succeeds in combining these features. In Chapter 5, we use the best of both worlds by defining an optimization problem that takes subpath constraints, minimizes node abundance errors, and is polynomially solvable. This establishes a theoretical novelty, of which the value in transcriptome assembly remains to be explored.

6.3.3 Haplotype reconstruction in tumors

Another application of great interest is haplotype reconstruction in tumor samples. Cancer cells are driven by mutations that cause cells to replicate and proliferate at much higher rates than healthy cells. Further mutations can occur as the tumor grows, leading to a heterogeneous population of haplotypes. Analysis of individual haplotypes within a tumor may contribute to a further understanding of tumor evolution and development of effective cancer treatments. Cancer genomes are often completely rearranged compared to the reference genome, making an important argument for de novo assembly.

Haplotype reconstruction in cancer genomes has much in common with the viral quasispecies assembly problem, the main difference being the genome size: $\sim 10^3$ bases for RNA virus genomes versus $\sim 3 \cdot 10^9$ bases for the human genome. Although the algorithms described in Chapter 2 do not scale well to such large genomes, the theory and techniques presented here may prove useful in this application as well. The assembly algorithm in Chapter 3, on the other hand, can process complete human chromosomes, but specializes in genomes of known ploidy. In the future, a combination of techniques from Chapters 2 and 3 may enable overlap graph-based haplotype assembly for cancer genomes.

6.4 Perspectives on third-generation sequencing

Sequencing technologies are evolving quickly and the long reads produced by TGS machines have led to significant advances in de novo haplotype assembly. In genomes of fixed ploidy (e.g. human and plant genomes) TGS has already made a great impact [53, 58, 61]. Long reads enable improved reconstruction of repeat-rich genomic regions, and allow bridging haplotypes across variant deserts. However, while this resolves existing problems, a new challenge arises: with sequencing error rates still above 10%, error correction becomes the main hurdle for any assembly algorithm.

In case of viral quasispecies, TGS technologies are able to cover an entire genome with a single read, thus transforming the assembly problem into a clustering problem. This does not necessarily make it an easier task, as high sequencing error rates lead to

even greater difficulties when constructing low-frequency strains. This area of research remains largely unexplored, with many problems still to be tackled. In order to apply techniques such as maximal clique enumeration in overlap graphs, not only substitution errors but also sequencing errors in the form of insertions and deletions need to be taken into account by the algorithms. We believe that adaptation of the approaches presented in this thesis could be a first step towards viral quasispecies reconstruction from TGS data.

Bibliography

[1] 1000 Genomes Project Consortium, Abecasis, G., Auton, A., Brooks, L., DePristo, M., Durbin, R., … McVean, G. (2012). An integrated map of genetic variation from 1,092 human genomes. *Nature*, *491*(7422), 56–65.

[2] Aguiar, D. & Istrail, S. (2012). HapCompass: a fast cycle basis algorithm for accurate haplotype assembly of sequence data. *Journal of Computational Biology*, *19*(6), 577–590.

[3] Ahn, S. & Vikalo, H. (2018). aBayesQR: a bayesian method for reconstruction of viral populations characterized by low diversity. *Journal of Computational Biology*, *25*(7), 637–648.

[4] Ahuja, R. K., Magnanti, T. L. & Orlin, J. B. (1993). *Network flows: theory, algorithms, and applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

[5] Altschul, S., Gish, W., Miller, W., Myers, E. & Lipman, D. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, *215*, 403–410.

[6] Astrovskaya, I., Tork, B., Mangul, S., Westbrooks, K., Mandoiu, I., Balfe, P. & Zelikovsky, A. (2011). Inferring viral quasispecies from 454 pyrosequencing reads. *BMC Bioinformatics*, *12*(Supp 1), S1.

[7] Baaijens, J. A., Zine El Aabidine, A., Rivals, E. & Schönhuth, A. (2017). De novo assembly of viral quasispecies using overlap graphs. *Genome Research*, *27*(5), 835–848.

[8] Baaijens, J., van der Roest, B., Köster, J., Stougie, L. & Schönhuth, A. (2019). Full-length de novo viral quasispecies assembly through variation graph construction. *Bioinformatics, to appear*.

[9] Bankevich, A., Nurk, S., Antipov, D., Gurevich, A., Dvorkin, M., Kulikov, A., … Alekseyev, M. (2012). SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology*, *19*(5), 455–477.

[10] Barik, S., Das, S. & Vikalo, H. (2018). Qsdpr: viral quasispecies reconstruction via correlation clustering. *Genomics*, *110*(6), 375–381.

[11] Beerenwinkel, N., Günthard, H. F., Roth, V. & Metzner, K. J. (2012). Challenges and opportunities in estimating viral genetic diversity from next-generation sequencing data. *Frontiers in Microbiology*, *3*, 239.

[12] Berger, E., Yorukoglu, D., Peng, J. & Berger, B. (2014). HapTree: a novel bayesian framework for single individual polyplotyping using ngs data. *PLOS Computational Biology*, *10*(3), 1–10.

[13] Bernard, E., Jacob, L., Mairal, J. & Vert, J. (2014). Efficient RNA isoform identification and quantification from RNA-Seq data with network flows. *Bioinformatics*, *30*(17), 2447–2455.

[14] Besenbacher, S., Liu, S., Izarzugaza, J. M. G., Grove, J., Belling, K., Bork-Jensen, J., … Rasmussen, S. (2015). Novel variation and de novo mutation rates in population-wide de novo assembled Danish trios. *Nature Communications*, *6*, 5969.

[15] Boisvert, S., Raymond, F., Godzaridis, E., Laviolette, F. & Corbeil, J. (2012). Ray meta: scalable de novo metagenome assembly and profiling. *Genome Biology*, *13*(12), R122.

[16] Bosman, T. (2015). A solution merging heuristic for the steiner problem in graphs using tree decompositions. In E. Bampis (Ed.), *Experimental algorithms* (pp. 391–402). Cham: Springer International Publishing.

[17] Bradnam, K. R., Fass, J. N., Alexandrov, A., Baranay, P., Bechner, M., Birol, I., … Korf, I. F. (2013). Assemblathon 2: evaluating de novo method of genome assembly in three vertebrate species. *GigaScience*, *2*, 10.

[18] Bray, N. L., Pimentel, H., Melsted, P. & Pachter, L. (2016). Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnology*, *34*, 525–527.

[19] Caskey, J., Wiseman, R., Karl, J., Baker, D., Lee, T., Raveendran, M., … O'Connor, D. (2019). MHC genotyping from rhesus macaque exome sequences. *bioRxiv*.

[20] Castel, S. E., Mohammadi, P., Chung, W. K., Shen, Y. & Lappalainen, T. (2016). Rare variant phasing and haplotypic expression from rna sequencing with phaser. *Nature Communications*, *7*.

[21] Chaisson, M. J., Sanders, A. D., Zhao, X., Malhotra, A., Porubsky, D., Rausch, T., … Lee, C. (2019). Multi-platform discovery of haplotype-resolved structural variation in human genomes. *Nature Communications*, *10*.

[22] Chen, J., Zhao, Y. & Sun, Y. (2018). De novo haplotype reconstruction in viral quasispecies using paired-end read guided path finding. *Bioinformatics*, bty202.

[23] Chin, C., Peluso, P., Sedlazeck, F. J., Nattestad, M., Concepcion, G. T., Clum, A., … Schatz, M. C. (2016). Phased diploid genome assembly with single molecule real-time sequencing. *Nature Methods*, *13*, 1050–1054.

[24] Choo, S. Y. (2007). The HLA system: genetics, immunology, clinical testing, and clinical implications. *Yonsei Medical Journal*, *48*(1), 11–23.

[25] Compeau, P., Pevzner, P. & Tesler, G. (2011). How to apply de Bruijn graphs to genome assembly. *Nature Biotechnology*, *29*, 987–991.

[26] Conesa, A., Madrigal, P., Tarazona, S., Gomez-Cabrero, D., Cervera, A., McPherson, A., … Mortazavi, A. (2016). A survey of best practices for RNA-seq data analysis. *Genome Biology*, *17*, 13.

[27] Cook, W. & Seymour, P. (2003). Tour merging via branch-decomposition. *INFORMS Journal on Computing*, *15*(3), 233–248.

[28] Crotty, S., Cameron, C. & Andino, R. (2001). RNA virus error catastrophe: direct molecular test by using ribavirin. *Proceedings of the National Academy of Sciences*, *98*(12), 6895–6900.

[29] Das, S. & Vikalo, H. (2015). SDhaP: haplotype assembly for diploids and polyploids via semi-definite programming. *BMC Genomics*, *16*(1), 260.

[30] Di Giallonardo, F., Töpfer, A., Rey, M., Prabhakaran, S., Duport, Y., Leemann, C., … Metzner, K. (2014). Full-length haplotype reconstruction to infer the structure of heterogeneous virus populations. *Nucleic Acids Research*, *42*, e115.

[31] Dilthey, A., Cox, C., Iqbal, Z., Nelson, M. & McVean, G. (2015). Improved genome inference in the MHC using a population reference graph. *Nature Genetics*, *47*, 682–688.

[32] Domingo, E., Sheldon, J. & Perales, C. (2012). Viral quasispecies evolution. *Microbiology and Molecular Biology Reviews*, *76*(2), 159–216.

[33] Dudley, D. M., Aliota, M. T., Mohr, E. L., Weiler, A. M., Lehrer-Brey, G., Weisgrau, K. L., … Friedrich, D. H., Thomas C. O/'Connor. (2016). A rhesus macaque model of Asian-lineage Zika virus infection. *Nature Communications*, *7*, 12204.

[34] Duffy, S., Shackelton, L. A. & Holmes, E. C. (2008). Rates of evolutionary change in viruses: patterns and determinants. *Nature Reviews Genetics*, *9*, 267–276.

[35] Duffy, S. (2018). Why are rna virus mutation rates so damn high? *PLOS Biology*, *16*(8), 1–6.

[36] Edge, P., Bafna, V. & Bansal, V. (2017). HapCUT2: robust and accurate haplotype assembly for diverse sequencing technologies. *Genome Research*, *27*(5), 801–812.

[37] EMBL-EBI. (2017). Vega genome browser. Retrieved September 28, 2017, from http://vega.archive.ensembl.org/info/data/MHC_Homo_sapiens.html

[38] Eppstein, D., Löffler, M. & Strash, D. (2010). Listing all maximal cliques in sparse graphs in near-optimal time. In *Proc. 21st int. symp. isaac* (Vol. 6506, pp. 403–414).

[39] Farci, P., Strazzera, R., Alter, H., Farci, S., Degioannis, D., Coiana, A., … Purcell, R. (2002). Early changes in hepatitis C viral quasispecies during interferon therapy predict the therapeutic outcome. *Proceedings of the National Academy of Sciences*, *99*(5), 3081–3086.

[40] Feng, J., Li, W. & Jiang, T. (2010). Inference of isoforms from short sequence reads. In B. Berger (Ed.), *Research in computational molecular biology* (pp. 138–157). Berlin, Heidelberg: Springer Berlin Heidelberg.

[41] Fischer, M., Strauch, B. & Renard, B. (2017). Abundance estimation and differential testing on strain level in metagenomics data. *Bioinformatics*, *33*(14), i124–i132.

[42] Garg, R., Shankar, R., Thakkar, B., Kudapa, H., Krishnamurthy, L., Mantri, N., … Jain, M. (2016). Transcriptome analyses reveal genotype- and developmental stage-specific molecular responses to drought and salinity stresses in chickpea. *Scientific Reports*, *6*, 19228.

[43] Garrison, E., Sirén, J., Novak, A., Hickey, G., Eizenga, J., Dawson, E., … Durbin, R. (2018). Variation graph toolkit improves read mapping by representing genetic variation in the reference. *Nature Biotechnology*, *36*, 875–879.

[44] Gill, S. R., Pop, M., DeBoy, R. T., Eckburg, P. B., Turnbaugh, P. J., Samuel, B. S., … Nelson, K. E. (2006). Metagenomic analysis of the human distal gut microbiome. *Science*, *312*(5778), 1355–1359.

[45] Glusman, G., Cox, H. & Roach, J. C. (2014). Whole-genome haplotyping approaches and genomic medicine. *Genome Medicine*, *6*(9), 73.

[46] Gregor, I., Schönhuth, A. & McHardy, A. (2016). Snowball: strain aware gene assembly of metagenomes. *Bioinformatics*, *32*(17), i649–i657.

[47] Gurevich, A., Saveliev, V., Vyahhi, N. & Tesler, G. (2013). QUAST: quality assessment tool for genome assemblies. *Bioinformatics*, *29*(8), 1072–1075.

[48] Hedegaard, D., Tully, D. C., Rowe, I. A., Reynolds, G. M., Bean, D. J., Hu, K., … McKeating, J. A. (2017). High resolution sequencing of hepatitis C virus reveals limited intra-hepatic compartmentalization in end-stage liver disease. *Journal of Hepatology*, *66*(1), 28–38.

[49] Hobbs, M., Pavasovic, A., King, A. G., Prentis, P. J., Eldridge, M. D., Chen, Z., … Timms, P. (2014). A transcriptome resource for the koala (Phascolarctos cinereus): insights into koala retrovirus transcription and sequence diversity. *BMC Genomics*, *15*(1), 786.

[50] Huang, A., Kantor, R., DeLong, A., Schreier, L. & Istrail, S. (2012). QColors: an algorithm for conservative viral quasispecies reconstruction from short and non-contiguous next generation sequencing reads. *In Silico Biology*, 193–201.

[51] Hunt, M., Gall, A., Ong, S. H., Brener, J., Ferns, B., Goulder, P., … Otto, T. D. (2015). IVA: accurate de novo assembly of RNA virus genomes. *Bioinformatics*, *31*(14), 2374–2376.

[52] Hunt, M., Newbold, C., Berriman, M. & Otto, T. D. (2014). A comprehensive evaluation of assembly scaffolding tools. *Genome Biology*, *15*(3), R42.

[53] Jain, M., Koren, S., Miga, K. H., Quick, J., Rand, A. C., Sasani, T. A., … Loose, M. (2018). Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature Biotechnology*, *36*, 338–345.

[54] Janeway, C. J., Travers, P., M., W. & M.J., S. (2001). *Immunobiology: the immune system in health and disease*. New York: Garland Science.

[55] Kajitani, R., Toshimoto, K., Noguchi, H., Toyoda, A., Ogura, Y., Okuno, M., … Itoh, T. (2014). Efficient de novo assembly of highly heterozygous genomes from whole-genome shotgun short read. *Genome Research*, *24*(8), 1384–1395.

[56] Kloster, K., Kuinke, P., O'Brien, M., Reidl, F., Villaamil, F. S., Sullivan, B. & van der Poel, A. (2017). A practical fpt algorithm for flow decomposition and transcript assembly. *CoRR*, *abs/1706.07851*.

[57] Knyazev, S., Tsyvina, V., Melnyk, A., Artyomenko, A., Malygina, T., Porozov, Y. B., … Zelikovsky, A. (2018). Cliquesnv: scalable reconstruction of intra-host viral populations from ngs reads. *bioRxiv*.

[58] Koren, S., Rhie, A., Walenz, B., Dilthey, A., Bickhart, D., Kingan, S., … Phillippy, A. (2018). De novo assembly of haplotype-resolved genomes with trio binning. *Nature Biotechnology*, *36*(12), 1174–1182. cited By 8.

[59] Kucherov, G. & Tsur, D. (2014). Improved filters for the approximate suffix-prefix overlap problem. In E. Moura & M. Crochemore (Eds.), *Proceedings of the 21st International Symposium on String Processing and Information Retrieval* (pp. 139–148). Cham: Springer International Publishing.

[60] Kuleshov, V. (2014). Probabilistic single-individual haplotyping. *Bioinformatics*, *30*(17), i379–85.

[61] Kyriakidou, M., Tai, H. H., Anglin, N. L., Ellis, D. & Strömvik, M. V. (2018). Current strategies of polyploid plant genome sequence assembly. *Frontiers in Plant Science*, *9*, 1660.

[62] Li, D., Liu, C.-M., Luo, R., Sadakane, K. & Lam, T.-W. (2015). MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics*, *31*(10), 1674–1676.

[63] Li, H. (2013). *Aligning sequence reads, clone sequences and assembly contigs with bwa-mem*. arXiv:1303.3997.

[64] Li, H. (2015). Microbiome, metagenomics, and high-dimensional compositional data analysis. *Annual Review of Statistics and Its Application*, *2*(1), 73–94.

[65] Li, H. & Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, *25*(14), 1754–1760.

[66] Li, H. (2018). Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, bty191.

[67] Li, W., Feng, J. & Jiang, T. (2011). Isolasso: a lasso regression approach to rna-seq based transcriptome assembly. *Journal of Computational Biology*, *18*, 1693–707.

[68] Lindner, M. & Renard, B. (2012). Metagenomic abundance estimation and diagnostic testing on species level. *Nucleic Acids Research*, *41*(1), e10.

[69] Luo, R., Liu, B., Xie, Y., Li, Z., Huang, W., Yuan, J., … Wang, J. (2012). Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience*, *1*(1), 18.

[70] Mäkinen, V., Belazzougui, D., Cunial, F. & Tomescu, A. (2015). *Genome-scale algorithm design: biological sequence analysis in the era of high-throughput sequencing*. Cambridge University Press.

[71] Malhotra, R., Mukhopadhyay, M., Poss, M. & Acharya, R. (2016). *A frame-based representation of genomic sequences for removing errors and rare variant detection in NGS data*. arXiv:1604.04803.

[72] Malhotra, R., Wu, S., Mukhopadhyay, M., Rodrigo, A., Poss, M. & Acharya, R. (2016). *Maximum likelihood de novo reconstruction of viral populations using paired end sequencing data*. arXiv:1502.04239.

[73] Mangul, S., Wu, N., Mancuso, N., Zelikovsky, A., Sun, R. & Eskin, E. (2014). Accurate viral population assembly from ultra-deep sequencing data. *Bioinformatics*, *30*, i329–i337.

[74] Marschall, T., Marz, M., Abeel, T., Dijkstra, L., Dutilh, B., Ghaffaari, A., … Schönhuth, A. (2018). Computational pan-genomics: status, promises and challenges. *Briefings in Bioinformatics*, *19*(1), 118–135.

[75] Marschall, T., Costa, I., Canzar, S., Bauer, M., Klau, G., Schliep, A. & Schönhuth, A. (2012). CLEVER: clique-enumerating variant finder. *Bioinformatics*, *28*(22), 2875–2882.

[76] Martin, M. (2011). Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal*, *17*(1), 10–12.

[77] Matzaraki, V., Kumar, V., Wijmenga, C. & Zhernakova, A. (2017). The MHC locus and genetic susceptibility to autoimmune and infectious diseases. *Genome Biology*, *18*(1), 76.

[78] Medvedev, P., Pham, S., Chaisson, M., Tesler, G. & Pevzner, P. (2011). Paired de bruijn graphs: a novel approach for incorporating mate pair information into genome assemblers. *Journal of Computational Biology*, *18*(11), 1625–1634.

[79] Medvedev, P., Fiume, M., Dzamba, M., Smith, T. & Brudno, M. (2010). Detecting copy number variation with mated short reads. *Genome Research*, *20*(11), 1613–1622.

[80] Meyerson, M., Gabriel, S. & Getz, G. (2010). Advances in understanding cancer genomes through second-generation sequencing. *Nature Reviews Genetics*, *11*, 685–696.

[81] Mezlini, A. M., Smith, E., Fiume, M., Buske, O., Savich, G., Shah, S., … Brudno, M. (2013). iReckon: simultaneous isoform discovery and abundance estimation from RNA-seq data. *Genome Research*, *23*(3), 519–29.

[82] Mikheenko, A., Saveliev, V. & Gurevich, A. (2016). Metaquast: evaluation of metagenome assemblies. *Bioinformatics*, *32*(7), 1088–1090.

[83] Myers, E. W. (1995). Toward simplifying and accurately formulating fragment assembly. *Journal of Computational Biology*, *2*(2), 275–290.

[84] Myers, E. W. (2016). A history of DNA sequence assembly. *it - Information Technology*, *58*, 126–132.

[85] Myers, E. W., Sutton, G. G., Delcher, A. L., Dew, I. M., Fasulo, D. P., Flanigan, M. J., … Venter, J. C. (2000). A whole-genome assembly of drosophila. *Science*, *287*(5461), 2196–2204.

[86] Nesterov, Y. & Nemirovskii, A. (1994). *Interior-point polynomial algorithms in convex programming*. SIAM.

[87] Nishant, K. T., Singh, N. & Alani, E. (2009). Genomic mutation rates: what high-throughput methods can tell us. *BioEssays*, *31*(9), 912–920.

[88] Novak, A., Garrison, E. & Paten, B. (2017). A graph extension of the positional Burrows–Wheeler transform and its applications. *Algorithms for Molecular Biology*, *12*(18).

[89] Nurk, S., Meleshko, D., Korobeynikov, A. & Pevzner, P. (2017). metaSPAdes: a new versatile metagenomic assembler. *Genome Research*, *27*(5), 824–834.

[90] Paten, B., Novak, A. M., Eizenga, J. M. & Garrison, E. (2017). Genome graphs and the evolution of genome inference. *Genome Research*, *27*(5), 665–676.

[91] Patterson, M., Marschall, T., Pisanti, N., Van Iersel, L., Stougie, L., Klau, G. & Schönhuth, A. (2015). WhatsHap: Weighted haplotype assembly for future-generation sequencing reads. *Journal of Computational Biology*, *22*(6), 498–509.

[92] Peng, Y., Leung, H., Yiu, S. & Chin, F. (2012). Meta-IDBA: a de novo assembler for metagenomic data. *Bioinformatics*, *27*(13), i94–i101.

[93] Pertea, M., Pertea, G., Antonescu, C., Chang, T., Mendell, J. & Salzberg, S. (2015). StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. *Nature Biotechnoly*, *33*, 290–295.

[94] Pevzner, P. A., Tang, H. & Waterman, M. S. (2001). An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences*, *98*(17), 9748–9753.

[95] Pirola, Y., Zaccaria, S., Dondi, R., Klau, G., Pisanti, N. & Bonizzoni, P. (2016). HapCol: accurate and memory-efficient haplotype assembly from long reads. *Bioinformatics*, *32*(11), 1610–1617.

[96] Porubsky, D., Garg, S., Sanders, A. D., Korbel, J. O., Guryev, V., Lansdorp, P. M. & Marschall, T. (2017). Dense and accurate whole-chromosome haplotyping of individual genomes. *Nature Communications*, *8*, 1293.

[97] Prabhakaran, S., Rey, M., Zagordi, O., Beerenwinkel, N. & Roth, V. (2014). HIV haplotype inference using a propagating dirichlet process mixture model. *IEEE Transactions on Computational Biology and Bioinformatics*, *11*(1), 182–191.

[98] Prosperi, M. C. F. & Salemi, M. (2012). QuRe: software for viral quasispecies reconstruction from next-generation sequencing data. *Bioinformatics*, *28*(1), 132–133.

[99] Quince, C., Lanzen, A., Davenport, R. J. & Turnbaugh, P. J. (2011). Removing noise from pyrosequenced amplicons. *BMC Bioinformatics*, *12*, 38.

[100] Ribeiro, F. J., Przybylski, D., Yin, S., Sharpe, T., Gnerre, S., Abouelleil, A., … Jaffe, D. (2012). Finished bacterial genomes from shotgun sequence data. *Genome Research*, *22*(11), 2270–2277.

[101] Rizzi, R., Tomescu, A. & Mäkinen, V. (2014). On the complexity of minimum path cover with subpath constraints for multi-assembly. *BMC Bioinformatics*, *15*(9), S5.

[102] Rose, R., Constantinides, B., Tapinos, A. & Robertson, D. (2016). Challenges in the analysis of viral metagenomes. *Virus Evolution*, *2*(2).

[103] Rosen, Y., Eizenga, J. & Paten, B. (2017). Modelling haplotypes with respect to reference cohort variation graphs. *Bioinformatics*, *33*(14), i118–i123.

[104] Safonova, Y., Bankevich, A. & Pevzner, P. (2015). DipSPAdes: Assembler for Highly Polymorphic Diploid Genomes. *Journal of Computational Biology*, *22*(6), 528–545.

[105] Salzberg, S., Phillippy, A., Zimin, A., Puiu, D., Magoc, T., Koren, S., … Yorke, J. (2011). GAGE: A critical evaluation of genome assemblies and assembly algorithms. *Genome Research*, *22*, 557–567.

[106] Sanger, F., Nicklen, S. & Coulson, A. (1977). DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, *74*(12), 5463–5467.

[107] Sczyrba, A., Hofmann, P., Belmann, P., Koslicki, D., Janssen, S., Dröge, J., … McHardy, A. (2017). Critical assessment of metagenome interpretation - a benchmark of metagenomics software. *Nature Methods*, *14*, 1063–1071.

[108] Shao, M. & Kingsford, C. (2017). Theory and a heuristic for the minimum path flow decomposition problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *PP*(99), 1–1.

[109] Shendure, J., Balasubramanian, S., Church, G. M., Gilbert, W., Rogers, J., Schloss, J. A. & Waterston, R. H. (2017). DNA sequencing at 40: past, present and future. *Nature*, *550*, 345–353.

[110] Simpson, J. & Durbin, R. (2012). Efficient de novo assembly of large genomes using compressed data structures. *Genome Research*, *22*, 549–556.

[111] Skums, P., Mancuso, N., Artyomenko, A., Tork, B., Mandoiu, I., Khudyakov, Y. & Zelikovsky, A. (2013). Reconstruction of viral population structure from next-generation sequencing data using multicommodity flows. *BMC Bioinformatics*, *14*(Suppl 9), S2.

[112] St. John, J. (2014). An illumina paired-end and mate-pair short read simulator. Retrieved February 12, 2015, from https://github.com/jstjohn/SimSeq

[113] Sudmant, P. H., Rausch, T., Gardner, E. J., Handsaker, R. E., Abyzov, A., Huddleston, J., ... Korbel, J. O. (2015). An integrated map of structural variation in 2504 human genomes. *Nature*, *526*, 75–81.

[114] Tewhey, R., Bansal, V., Torkamani, A., Topol, E. & Schork, N. (2011). The importance of phase information for human genomics. *Nature Reviews Genetics*, *12*(3), 215.

[115] The Genome of the Netherlands Consortium. (2014). Whole-genome sequence variation, population structure and demographic history of the dutch population. *Nature Genetics*, *46*, 818–825.

[116] The UK10K Consortium. (2015). The UK10K project identifies rare variants in health and disease. *Nature*, *526*, 82–90.

[117] Tomescu, A. I., Kuosmanen, A., Rizzi, R. & Mäkinen, V. (2013). A novel min-cost flow method for estimating transcript expression with RNA-Seq. *BMC Bioinformatics*, *14*(5), S15.

[118] Töpfer, A., Marschall, T., Bull, R., Luciani, F., Schönhuth, A. & Beerenwinkel, N. (2014). Viral quasispecies assembly via maximal clique enumeration. *PLoS Computational Biology*, *10*(3), e1003515.

[119] Töpfer, A., Zagordi, O., Prabhakaran, S., Roth, V., Halperin, E. & Beerenwinkel, N. (2013). Probabilistic inference of viral quasispecies subject to recombination. *Journal of Computational Biology*, *20*(2), 113–123.

[120] Trapnell, C., Williams, B., Pertea, G., Mortazavi, A., Kwan, G., van Baren, M., ... Pachter, L. (2010). Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, *28*, 511–515.

[121] Välimäki, N., Ladra, S. & Mäkinen, V. (2012). Approximate all-pairs suffix/prefix overlaps. *Information and Computation*, *213*, 49–58.

[122] Vatinlen, B., Chauvet, F., Chrétienne, P. & Mahey, P. (2008). Simple bounds and greedy algorithms for decomposing a flow into a minimal set of paths. *European Journal of Operational Research*, *185*(3), 1390–1401. cited By 19.

[123] Venter, J. C., Remington, K., Heidelberg, J. F., Halpern, A. L., Rusch, D., Eisen, J. A., ... Smith, H. O. (2004). Environmental genome shotgun sequencing of the sargasso sea. *Science*, *304*(5667), 66–74.

[124] Victoria Wang, X., Blades, N., Ding, J., Sultana, R. & Parmigiani, G. (2012). Estimation of sequencing error rates in short reads. *BMC Bioinformatics*, *13*(1), 185.

[125] Vignuzzi, M., Stone, J., Arnold, J., Cameron, C. & Andino, R. (2006). Quasispecies diversity determines pathogenesis through cooperative interactions in a viral population. *Nature*, *439*, 344–348.

[126] Weirather, J., de Cesare, M., Wang, Y., Piazza, P., Sebastiano, V., Wang, X. J., ... Au, K. F. (2017). Comprehensive comparison of Pacific Biosciences and Oxford Nanopore Technologies and their applications to transcriptome analysis. *F1000Research*, *6*, 100.

[127] Weisenfeld, N. I., Kumar, V., Shah, P., Church, D. & Jaffe, D. (2017). Direct determination of diploid genome sequences. *Genome Research*, *27*, 757–767.

[128] Xie, M., Wu, Q., Wang, J. & Jiang, T. (2016). H-PoP and H-PoPG: heuristic partitioning algorithms for single individual haplotyping of polyploids. *Bioinformatics*, *32*(24), 3735–3744.

[129] Yang, X., Charlebois, P., Gnerre, S., Coole, M., Lennon, N., Levin, J., ... Henn, M. (2012). De novo assembly of highly diverse viral populations. *BMC Genomics*, *13*(1), 475.

[130] Zagordi, O., Bhattacharya, A., Eriksson, N. & Beerenwinkel, N. (2011). ShoRAH: estimating the genetic diversity of a mixed sample from next-generation sequencing data. *BMC Bioinformatics*, *12*(1), 119.

[131] Zagordi, O., Geyrhofer, L., Roth, V. & Beerenwinkel, N. (2010). Deep sequencing of a genetically heterogeneous sample: local haplotype reconstruction and read error correction. *Journal of Computational Biology*, *17*(3), 417–428.

[132] Zhang, J., Kobert, K., Flouri, T. & Stamatakis, A. (2014). PEAR: A fast and accurate Illumina Paired-End reAd mergeR. *Bioinformatics*, *30*, 614–620.

Summary

Many genomes come in copies, where each copy stems from one of the ancestors. The number of copies determines the *ploidy* of the organism: *haploid* denotes a single copy, while *diploid* relates to two copies, and *polyploid* refers to more than two copies (depending on the context, polyploid may also include diploid). For example, the human genome is diploid, encoded as DNA within 46 chromosomes which come in 23 pairs, one copy from each parent.

Genomes within a population show *genetic variation* as a result of mutation and recombination. Also the copies of the genome in a single individual will differ in terms of the genetic variants affecting them. These copy-specific sequences are referred to as *haplotypes*. The genetic differences between individuals play an important role in evolution, as genomic alterations can affect gene expression levels and enable the development of novel gene functions.

Sequencing technologies enable reading genomic sequences, but only for relatively short pieces of sequence, called *reads*. The majority of sequencing machines installed worldwide perform so-called *next-generation sequencing* (NGS) and have filled up databases with these traditional, short NGS reads. The goal of *haplotype-aware genome assembly* is to reconstruct each of the individual haplotypes from a given set of sequencing reads, such as a human genome (diploid), a potato plant genome (tetraploid), or a mixture of genetically related virus strains in an infection ("viral quasispecies", unknown ploidy). Haplotype-aware genome assembly is an important step in genetics, medicine, and various other disciplines.

However, generation of haplotype-resolved de novo assemblies is a major challenge. Beyond distinguishing between errors and true sequential variants, one needs to assign the true variants to the different genome copies. Reference-genome-independent ("de novo") approaches have yielded benefits over reference-guided approaches, because reference-induced biases can have a great impact on assembly quality when dealing with divergent haplotypes. We present several new approaches to de novo assembly of individual haplotypes from mixed samples.

After a brief introduction to the field of haplotype-aware genome assembly in

Chapter 1, we explore the possibilities of *overlap graph-based the novo assembly* in **Chapters 2 and 3**. In these overlap graphs, nodes represent reads, while edges reflect that two reads represent identical haplotypic sequence (based on sound statistical considerations). We introduce two new overlap assembly algorithms, each following an iterative scheme where reads or contigs are joined in such a way that contigs grow while preserving their haplotype identity. **Chapter 2** focuses on viral quasispecies assembly: we use overlap graphs for accurate reconstruction of viral haplotypes within an infection. In **Chapter 3**, on the other hand, we develop an overlap graph-based assembly algorithm for a scenario of low sequencing depth where the ploidy of the genome is known.

Both chapters highlight the strength of overlap graph-based assembly when aiming to distinguish between haplotypes. While assembling a consensus sequence usually works favorably based on data structures known as *de Bruijn graphs*, this requires sequencing reads to be decomposed into k-mers, where k is usually considerably smaller than the read length. We point out that using reads at their full length is key in assembling individual haplotypes; examining the full read span enhances the correction of sequencing errors and the reconstruction of low-frequency haplotypes. In addition, we illustrate the advantages of de novo assembly over reference-guided approaches in various benchmarking experiments.

Chapter 4 continues the search for a de novo solution to the viral quasispecies assembly problem. Although **Chapter 2** presents a solution to the core problem, the assembly of strain-specific contigs, these contigs do not yet reflect full-length viral haplotypes. In a viral quasispecies, each haplotype can appear at a different frequency, with some strains being highly abundant while others are very rare. An important component of the viral quasispecies assembly problem is the estimation of relative abundances for the reconstructed haplotypes. In **Chapter 4**, we extend the contigs obtained previously into full-length haplotypes *and* compute haplotype abundances, by making use of variation graph-based principles and defining an appropriate optimization problem.

Finally, in **Chapter 5** we resolve the remaining practical problems regarding full-length viral quasispecies reconstruction. With increasing genome sizes and increasing complexity of the data set, the solution in the previous chapter experience serious limitations. We overcome these computational issues by reformulating the mathematical problem, allowing for more efficient optimization techniques. This chapter completes our solution to the de novo viral quasispecies assembly problem; the mathematical framework presented here has the potential to make a big step ahead in haplotype-aware genome assembly in general.

Overall, this thesis presents the first de novo approach to full-length viral quasispecies assembly, as well as a novel solution to the haplotype-aware assembly of genomes of known ploidy. These methods have proved valuable in many applica-

tions, among which the analysis of viral infections from patient samples (e.g. Zika virus, human immunodeficiency virus (HIV), Ebola virus, and hepatitis C virus) and the accurate reconstruction of heavily divergent regions in long genomes (such as the MHC region in human genomes).

Samenvatting

Algoritmen ten behoeve van haplotype assemblage zonder referentiegenoom

Vaak bestaat een genoom uit meerdere kopieën, waarbij iedere kopie van één van de directe voorouders komt. Het aantal kopieën bepaalt de *ploïdiegraad* van het organisme: *haploïdie* betekent slechts één kopie, *diploïdie* betekent twee kopieën en *polyploïdie* betekent meer dan twee kopieën (afhankelijk van de context valt diploïdie in sommige gevallen ook onder polyploïdie). Ter illustratie, het menselijk genoom is diploïde, gecodeerd in de vorm van DNA verdeeld over 46 chromosomen; deze chromosomen vormen 23 paren, waarvan één kopie van de vader en één kopie van de moeder komt.

Door wijzigingen van het erfelijk materiaal (mutaties) onstaat er binnen een populatie *genetische variatie*. Ook de kopieën binnen een specifiek genoom verschillen genetisch gezien van elkaar. Deze verschillende kopieën worden *haplotypen* genoemd. De genetische verschillen tussen individuen spelen een belangrijke rol binnen de evolutie, omdat ze kunnen zorgen voor veranderingen in eigenschappen en het ontwikkelen van nieuwe functies.

Met behulp van *sequencing* kunnen we het erfelijk materiaal uitlezen. Dit werkt echter alleen voor stukken van beperkte lengte, ook wel *reads* genaamd. Het merendeel van alle sequencing machines wereldwijd maakt gebruik van *next-generation sequencing* (NGS) technieken, gekenmerkt door relatief korte reads. Het doel van *haplotype assemblage* is om de individuele haplotypen te reconstrueren vanuit de sequencing reads. Bijvoorbeeld de twee haplotypen van een menselijk genoom, de vier haplotypen van het genoom van een aardappelplant, of alle verschillende stammen van een virus binnen het lichaam van een patiënt (een virale quasi-soort). Haplotype assemblage is een belangrijk onderdeel van de genetica, de geneeskunde en andere disciplines.

Het is echter een grote uitdaging om haplotypen te reconstrueren zonder gebruik te maken van een referentiegenoom. Dit vereist dat er onderscheid gemaakt wordt tussen fouten van de sequencing machine en werkelijke genetische variatie. Bovendien moeten de verschillende varianten toebedeeld worden aan de verschillende haplotypen. Omdat het gebruik van een referentiegenoom onzuiverheden in het resultaat kan veroorzaken, presenteren we hier verschillende referentievrije (“de novo”) methoden ten behoeve

van haplotype assemblage.

Dit proefschrift begint met een korte introductie tot het onderzoek rondom haplotype assemblage in **hoofdstuk 1**. Vervolgens gaan we in **hoofdstuk 2 en 3** in op de mogelijkheden tot het gebruik van een *overlap graaf* bij de novo haplotype assemblage. Dit type graaf bevat een knoop voor iedere read in de dataset en heeft een gerichte zijde tussen twee knopen wanneer de bijbehorende reads, op basis van statistische overwegingen, tot hetzelfde haplotype behoren. We beschrijven twee nieuwe algoritmen die beiden gebruik maken van deze overlap graaf. Hierin worden herhaaldelijk sequenties samengevoegd tot langere stukken *(contigs)*, zodanig dat deze specifiek blijven voor één haplotype. In **hoofdstuk 2** richten we ons op het reconstrueren van een virale quasi-soort met behulp van een overlap graaf. In **hoofdstuk 3** ontwikkelen we vervolgens een algoritme dat zich richt op organismen waarvan de ploïdiegraad wél bekend is (in tegenstelling tot virale quasi-soorten).

Beide hoofdstukken laten zien hoe waardevol een overlap graaf is wanneer je individuele haplotypen van elkaar wilt onderscheiden. Voor het reconstrueren van een consensus genoom (in plaats van individuele haplotypen) is een zogenaamde *de Bruijn graaf* over het algemeen een efficiënt alternatief. Hierbij worden alleen alle deelreeksen van de reads van lengte k (de k-mers) opgeslagen. Wij laten zien dat het gebruik van de volledige reads cruciaal is vanuit het oogpunt van haplotype assemblage, omdat dit helpt bij het corrigeren van fouten gemaakt door de sequencing machine. Hierdoor kunnen we ook haplotypen reconstrueren die erg zeldzaam zijn binnen de populatie. Tenslotte laten we via verschillende experimenten het voordeel van een referentievrije aanpak zien ten opzichte van referentiegestuurde methoden.

In **hoofdstuk 4** wordt de zoektocht naar een de novo oplossing voor het reconstrueren van virale quasi-soorten vervolgd. In **hoofdstuk 2** is al aangetoond hoe reads samengevoegd kunnen worden tot contigs van maximale lengte, zodanig dat iedere contig specifiek is voor een bepaald haplotype. Deze contigs zijn echter nog niet van dezelfde lengte als het genoom, ze moeten nog verder verlengd worden. In een virale quasi-soort komen de verschillende haplotypen ook in verschillende verhoudingen voor: in een infectie komen sommige stammen van het virus veel voor, terwijl andere erg zeldzaam zijn. Een belangrijk deel van het probleem rondom virale quasi-soorten is ook het schatten van deze verhoudingen. In **hoofdstuk 4** continueren we de assemblage zodat iedere contig een volledig haplotype weergeeft, terwijl we ook de verhoudingen van de stammen bepalen. Dit doen we met behulp van technieken gebaseerd op *variatie grafen*, waarbij we een geschikt optimalisatieprobleem definiëren.

Tenslotte nemen we in **hoofdstuk 5** de laatste stap tot het efficiënt en volledig reconstrueren van een virale quasi-soort. De methoden uit het vorige hoofdstuk zijn theoretisch afdoende, maar in de praktijk mogelijk problematisch vanwege de complexiteit van de oplossing. In dit hoofdstuk herformuleren we het combinatorische

probleem, zodat deze een polynomiale oplossing toestaat. Daarmee is onze referentievrije oplossing voor het reconstrueren van een virale quasi-soort compleet. Bovendien is deze oplossing dusdanig efficiënt dat ook grotere genomen tot de mogelijkheden behoren.

Samengevat presenteert dit proefschrift de eerste referentievrije methode tot volledige assemblage van virale quasi-soorten, alsook een oplossing voor het reconstrueren van haplotypen wanneer de ploïdiegraad bekend is. Deze methoden zijn waardevol in verschillende toepassingen, waaronder het analyseren van virale infecties (bijvoorbeeld Zika, HIV, Ebola en hepatitis C) en het reconstrueren van regionen in het menselijk genoom gekenmerkt door een grote genetische diversiteit, zoals het major histocompatibility complex.

Publications

- **J.A. Baaijens**, L. Stougie, and A. Schönhuth (2019). Strain-aware assembly of genomes from mixed samples using variation graphs. *Submitted manuscript.*

- **J.A. Baaijens**, B. van der Roest, J. Köster, L. Stougie, and A. Schönhuth (2019). Full-length de novo viral quasispecies assembly through variation graph construction. *Bioinformatics*, btz443 (in press).

- **J.A. Baaijens** and A. Schönhuth (2019). Overlap graph-based generation of haplotigs for diploids and polyploids. *Bioinformatics*, btz255 (in press).

- T. Marschall, M. Marz, T. Abeel, L. Dijkstra, B.E. Dutilh, A. Ghaffaari, P. Kersey, W.P. Kloosterman, V. Mäkinen, A.M. Novak, B. Paten, D. Porubsky, E. Rivals, C. Alkan, **J.A. Baaijens**, P.I.W. De Bakker, V. Boeva, R.J.P. Bonnal, F. Chiaromonte, R. Chikhi, F.D. Ciccarelli, R. Cijvat, E. Datema, C.M. Van Duijn, E.E. Eichler, C. Ernst, E. Eskin, E. Garrison, M. El-Kebir, G.W. Klau, J.O. Korbel, E.W. Lameijer, B. Langmead, M. Martin, P. Medvedev, J.C. Mu, P. Neerincx, K. Ouwens, P. Peterlongo, N. Pisanti, S. Rahmann, B. Raphael, K. Reinert, D. de Ridder, J. de Ridder, M. Schlesner, O. Schulz-Trieglaff, A.D. Sanders, S. Sheikhizadeh, C. Shneider, S. Smit, D. Valenzuela, J. Wang, L. Wessels, Y. Zhang, V. Guryev, F. Vandin, K. Ye, A. Schönhuth (2018). Computational pan-genomics: status, promises and challenges. *Briefings in Bioinformatics*, **19**(1):118–135.

- **J.A. Baaijens**, A. Zine El Aabidine, E. Rivals, and A. Schönhuth (2017). De novo assembly of viral quasispecies using overlap graphs. *Genome Research*, 27(5):835–848.

- J.Y. Hehir-Kwa, T. Marschall, W.P. Kloosterman, L.J. Dijkstra, **J.A. Baaijens**, L.J. Dijkstra, A. Abdellaoui, V. Koval, D.T. Thung, R. Wardenaar, I. Renkens, B.P. Coe, P. Deelen, J. de Ligt, E. Lameijer, F. van Dijk, F. Hormozdiari, The Genome of the Netherlands Consortium, A.G. Uitterlinden, C.M. van Duijn, E.E. Eichler, P.I.W. de Bakker, M.A. Swertz, C. Wijmenga, G.B. van Ommen, P.E. Slagboom, D.I. Boomsma, A. Schönhuth, K. Ye, V. Guryev (2016). A high-quality human

reference panel reveals the complexity and distribution of genomic structural variants. *Nature Communications*, 7:12989.

- **J.A. Baaijens** and J. Draisma (2016). On the existence of identifiable reparametrizations for linear compartment models. *SIAM Journal on Applied Mathematics*, 76(4):1577–1605.

- **J.A. Baaijens** and J. Draisma (2015). Euclidean distance degrees of real algebraic groups. *Linear Algebra and its Applications*, 467:174–187.

Selected conference presentations

- De novo viral quasispecies assembly using overlap graphs (highlight talk). *Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, Paris, France, 2018.

- Full-length de novo viral quasispecies assembly through variation graph construction. *RECOMB Satellite Workshop on Massively Parallel Sequencing (RECOMB-Seq)*, Paris, France, 2018.

- De novo viral quasispecies assembly using overlap graphs (highlight talk). *Conference on Intelligent Systems for Molecular Biology (ISMB)*, Prague, Czech Republic, 2017. ***Best talk award***.

- De novo viral quasispecies assembly using overlap graphs. *Dutch Bioinformatics & Systems Biology Conference (BioSB)*, Lunteren, the Netherlands, 2017.

- De novo assembly of viral quasispecies (poster presentation). *Annual International Conference on Research in Computational Molecular Biology (RECOMB)*, Los Angeles, U.S.A., 2016.

Curriculum vitae

Jasmijn Anne Baaijens was born in Alphen aan den Rijn, the Netherlands, on the 10th of December, 1990. In 2008, she started her studies in mathematics at Utrecht University, the Netherlands. During this time she also enjoyed organizing many student activities through the study association A-Eskwadraat. Besides her passion for mathematics, Jasmijn always had an interest in biology; thus, in 2010, she spent a semester at Université Paul Sabatier in Toulouse, France, to follow several courses in biology.

After obtaining her bachelor's degree (cum laude) in February 2012, Jasmijn decided to continue on a more applied course. She enrolled in the master's programme Industrial and Applied Mathematics at Eindhoven University of Technology (TUe), in the Discrete Mathematics and Applications track. She conducted two extra-curricular research projects as part of the honours programme: first, on secure multiparty computation of a logarithmic function, under the supervision of dr. Berry Schoenmakers; and second, on euclidean distance degrees of real algebraic groups, with prof.dr. Jan Draisma. She continued the latter collaboration for her graduation project. In 2014, she presented her master's thesis "On the existence of identifiable reparametrizations for compartment models" and obtained her master's degree, cum laude.

Immediately after, Jasmijn started her PhD in bioinformatics at CWI, under the supervision of prof.dr. Alexander Schönhuth. During her time as a PhD student, she got the opportunity to attend and present her work at several international conferences and workshops, the highlight being ISMB 2017 in Prague: there, Jasmijn won a "best talk award". Besides attending workshops, she also co-organized the 2017 edition of the workshop Data Structures in Bioinformatics (DSB) in Amsterdam. Subsequently, Jasmijn was invited to join the DSB steering committee and thus remains associated with this workshop. In addition to research and related activities, she spent her time at CWI organizing social events for trainees, PhD's, and postdocs, as a member and chair of the PhD Activity Committee. Jasmijn currently lives in Alphen aan den Rijn with her husband Michiel and her daughter Senna.