ABSTRACT

Title of dissertation: GENOME ASSEMBLY AND VARIANT DETECTION USING EMERGING SEQUENCING TECHNOLOGIES AND GRAPH BASED METHODS

Jay Ghurye
Doctor of Philosophy, 2018

Dissertation directed by: Professor Mihai Pop
Department of Computer Science

The increased availability of genomic data and the increased ease and lower costs of DNA sequencing have revolutionized biomedical research. One of the critical steps in most bioinformatics analyses is the assembly of the genome sequence of an organism using the data generated from the sequencing machines. Despite the long length of sequences generated by third-generation sequencing technologies (tens of thousands of basepairs), the automated reconstruction of entire genomes continues to be a formidable computational task. Although long read technologies help in resolving highly repetitive regions, the contigs generated from long read assembly do not always span a complete chromosome or even an arm of the chromosome. Recently, new genomic technologies have been developed that can "bridge" across repeats or other genomic regions that are difficult to sequence or assemble and improve genome assemblies by "scaffolding" together large segments of the genome. The problem of scaffolding is vital in the context of both single genome assembly of large eukaryotic genomes and in metagenomics where the goal is to assemble multiple bacterial genomes in a sample simultaneously.

First, we describe SALSA2, a method we developed to use interaction frequency between any two loci in the genome obtained using Hi-C technology to scaffold fragmented eukaryotic genome assemblies into chromosomes. SALSA2 can be used with either short or long read assembly to generate highly contiguous and accurate chromosome level assemblies. Hi-C data are known to introduce small inversion errors in the assembly, so we included the assembly graph in the scaffolding process and used the sequence overlap information to correct the orientation errors.

Next, we present our contributions to metagenomics. We developed a scaffolding and variant detection method "MetaCarvel" for metagenomic datasets. Several factors such as the presence of inter-genomic repeats, coverage ambiguities, and polymorphic regions in the genomes complicate the task of scaffolding metagenomes. Variant detection is also tricky in metagenomes because the different genomes within these complex samples are not known beforehand. We showed that MetaCarvel was able to generate accurate scaffolds and find genome-wide variations *de novo* in metagenomic datasets.

Finally, we present EDIT, a tool for clustering millions of DNA sequence fragments originating from the highly conserved 16s rRNA gene in bacteria. We extended classical Four Russians' speed up to banded sequence alignment and showed that our method clusters highly similar sequences efficiently. This method can also be used to remove duplicates or near duplicate sequences from a dataset.

With the increasing data being generated in different genomic and metagenomic studies using emerging sequencing technologies, our software tools and algorithms are well timed with the need of the community.

GENOME ASSEMBLY AND VARIANT DETECTION USING EMERGING SEQUENCING TECHNOLOGIES AND GRAPH BASED METHODS

by

Jay Ghurye

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Advisory Committee:
Professor Mihai Pop, Chair/Advisor
Dr. Adam M. Phillippy,
Professor Héctor Corrada Bravo,
Professor Aravind Srinivasan,
Professor Max Leiserson,
Professor Michael Cummings, Dean's Representative

© Copyright by
Jay Ghurye
2018

Preface

The algorithms, software, and results in this dissertation have either been published in peer-reviewed journals and conferences or are currently under preparation for submission. At the time of this writing, Chapters 2, 4, 5, and 7 have already been published or submitted for publication and are reformatted here. Chapter 3 and 6 are under preparation for submission. I am indebted to my co-authors on these projects - their dedication and knowledge in the areas of computer science, statistics, and biology have resulted in much stronger scientific papers.

- **Chapter 1**

  - **Jay Ghurye**, Victoria Cepeda-Espinoza, and Mihai Pop. "Focus: Microbiome: Metagenomic Assembly: Overview, Challenges and Applications." The Yale journal of biology and medicine 89.3 (2016): 353.

  My contributions to these works include (1) surveying the literature for recent and seminal results, and (2) writing the manuscripts.

- **Chapter 2**

  - **Jay Ghurye** and Mihai Pop. "Algorithms and technologies for uncovering genome structure," Under review.

  My contributions to these works include (1) surveying the literature for recent and seminal results, and (2) writing the manuscripts.

- **Chapter 3**

  - **Jay Ghurye**, Mihai Pop, Sergey Koren, Derek Bickhart, and Chen-Shan Chin.

"Scaffolding of long read assemblies using long range contact information." BMC genomics 18, no. 1 (2017): 527.

- **Jay Ghurye**, Arang Rhie, Brian P. Walenz, Anthony Schmitt, Siddarth Selvaraj, Mihai Pop, Adam M. Phillippy, and Sergey Koren. "Integrating Hi-C links with assembly graphs for chromosome-scale assembly." bioRxiv (2018): 261149.

My contributions to these works include (1) design and implementation of the algorithm, (2) doing software evaluation, and (3) writing the manuscript.

- **Chapter 4**

  - **Jay Ghurye**, Sergey Koren, Arang Rhie, Brian Walenz, Siddarth Selvaraj, Anthony Schmitt and Adam Phillippy "Effect of different Hi-C library preparation on eukaryotic genome scaffolding", Under preparation

  My contributions to these works include (1) designing and running experiments, (2) writing the manuscript.

- **Chapter 5**

  - **Jay Ghurye**, and Mihai Pop. "Better Identification of Repeats in Metagenomic Scaffolding." In International Workshop on Algorithms in Bioinformatics, pp. 174-184. Springer, Cham, 2016.

  My contributions to these works include (1) design and implementation of the algorithm, (2) doing software evaluation, (3) writing the manuscript.

- **Chapter 6**

  - **Jay Ghurye**, Todd Treangen, Sergey Koren, Marcus Fedarko, W. Judson Hervey

IV and Mihai Pop "MetaCarvel: linking assembly graph motifs to biological variants", Under preparation.

My contributions to these works include (1) design and implementation of the algorithm, (2) doing software evaluation, and (3) writing the manuscript.

- **Chapter 7**

  - Brian Brubach*, **Jay Ghurye***, Mihai Pop and Aravind Srinivasan, 2017. "Better Greedy Sequence Clustering with Fast Banded Alignment". In LIPIcs-Leibniz International Proceedings in Informatics (Vol. 88). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. (* denotes equal contribution)

  My contributions to these works include (1) design and implementation of the algorithm, (2) doing software evaluation, and (3) writing the manuscript.

Here is the list of other publications I have been involved, either as a first author or a contributing author.

- **Jay Ghurye**, Gautier Krings and Vanessa Frias-Martinez, 2016, June. "A Framework to Model Human Behavior at Large Scale during Natural Disasters". In Mobile Data Management (MDM), 2016 17th IEEE International Conference on (Vol. 1, pp. 18-27). IEEE.

- Brian Brubach*, **Jay Ghurye***, 2018. "A Succinct Four Russian's Speedup for Edit Distance Computation and One-against-many Banded Alignment". In LIPIcs-Leibniz International Proceedings in Informatics (Vol. 105). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (* denotes equal contribution).

- **Jay Ghurye**, Sergey Koren, Scott Small, Adam Phillippy, and Nora Besansky. "*De novo* assembly of *Anopheles funestus* genome". (Under preparation)

- - Marcus Fedarko, **Jay Ghurye**, Todd Treagen, and Mihai Pop. "MetagenomeScope: Web-Based Hierarchical Visualization of Metagenome Assembly Graphs." (2017): 630-632. (Under submission)

- Nathan D. Olson, Todd J. Treangen, Christopher M. Hill, Victoria Cepeda-Espinoza, **Jay Ghurye**, Sergey Koren, and Mihai Pop. "Metagenomic assembly through the lens of validation: recent advances in assessing and improving the quality of genomes assembled from metagenomes." Briefings in bioinformatics (2017).

- Seth Commichaux, Nidhi Shah, Alexander Stoppel, **Jay Ghurye**, Michael Cummings and Mihai Pop. "A Critical Analysis of the Integrated Gene Catalog," (Under preparation)

Dedication

To my parents for their unconditional love and support

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been the one that I will cherish forever.

First, I would like to thank my advisor, Professor Mihai Pop for giving me an invaluable opportunity to work on challenging and exciting projects. It has been a pleasure to work with and learn from such an extraordinary individual. His advice has always helped me at various stages in the graduate program and helped me grow as a researcher and better person.

I would also like to thank all of my other committee members - Michael Cummings, Aravind Srinivasan, Héctor Corrada Bravo, Max Leiserson, and Adam Phillippy for being interested in my research. Your feedback has been instrumental in shaping this dissertation.

I would like to thank all my collaborators who contributed to the work described in this dissertation. Primarily, I would want to thank Adam Phillippy for giving me an opportunity to work on exciting projects with his group at NIH. A special thank you to Sergey Koren and Arang Rhie for continuous guidance regarding genome assembly related concepts. I also especially want to thank Jason Chin, who was my mentor at Pacific Biosciences during my internship. Working with all of you has broadened my interest and knowledge in the area of Bioinformatics and Computational Biology.

I am very grateful to Professor Rajiv Gandhi without whom I would not have even considered pursuing a Ph.D. Because of his guidance, I felt confident to apply for graduate programs after my undergrad.

My colleagues at the Pop lab have enriched my graduate life in several ways. I am thankful to Victoria Cepeda, Brian Brubach, Nidhi Shah, Brook Stacy, Kiran Javkar, Jackie Meisel, Dan Nasko, Todd Treangen, Marcus Fedarko, Nate Olson, and Jeremy Selengut. I will miss the discussions we had over the lab lunches and in the offices. I am grateful to CBCB coordinators Christine Maria Bogan and Barbara Lewis for taking care of all the logistics related to conference travel. Also, I am thankful to UMIACS staff for providing and supporting excellent compute infrastructure.

I would like to thank my friends Yogarshi, Karthik, Jaideep, Suraj, Pallabi, Sudha, Mary, and many others for making my non-academic life enjoyable and memorable.

Finally, special thanks must go to my parents Vaibhavi and Shrirang, my sister Shruti, and my girlfriend Nidhi for their love and support. I appreciate it more than I can say.

It is impossible to remember everyone, so I apologize to anyone who may have been left out. If I forgot you, let me know, and I will happily buy you lunch.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

WGS	Whole Genome Sequencing
HMP	Human Microbiome Project
VGP	Vertebrate Genome Project
NGS	Next Generation Sequencing
FRC	Feature Response Curve
DNA	Deoxyribonucleic Acid
OTU	Operational Taxonomic Unit
STS	Sequence Tagged Sites
FISH	Fluorescent in situ hybridization
C3	Chromosomal Conformation Capture
BAC	Bacterial Artificial Chromosome
TSLR	TruSeq Synthetic Long-Read
TAD	Topologically Associated Domains
PCR	Polymerase Chain Reaction
GFA	Graphical Fragment Assembly
PCR	Polymerase Chain Reaction

Chapter 1: Introduction

The genome sequence of an organism is a blueprint to build that organism. It is a starting point that allows researchers to study an organism's evolutionary history and link its genomic features to phenotypic features and more. The invention of DNA sequencing technologies has revolutionized genome analysis. In 2001, by using sequencing data and sophisticated computational algorithms, the first draft sequences of the human genome were made publicly available [1, 2]. Owing to the fast-paced development of sequencing technologies and data analysis algorithms, the human genome has undergone at least a dozen revisions. Also, due to ease of genome sequencing, different plant and animal species are being sequenced to understand the biology of different species on the planet better.

The data generated by sequencing machines is fragmented and usually contains sequencing errors. Thus, significant computational efforts are involved in correcting the data and assembling it into a complete genome sequence. This need of computational methods has given rise to a whole new branch of bioinformatics called "genome assembly," where the goal is to develop computational methods to obtain complete genome sequences of an organism by using a combination of different sequencing technologies. Although the problem of genome assembly has been studied for about 40 years [3], it is

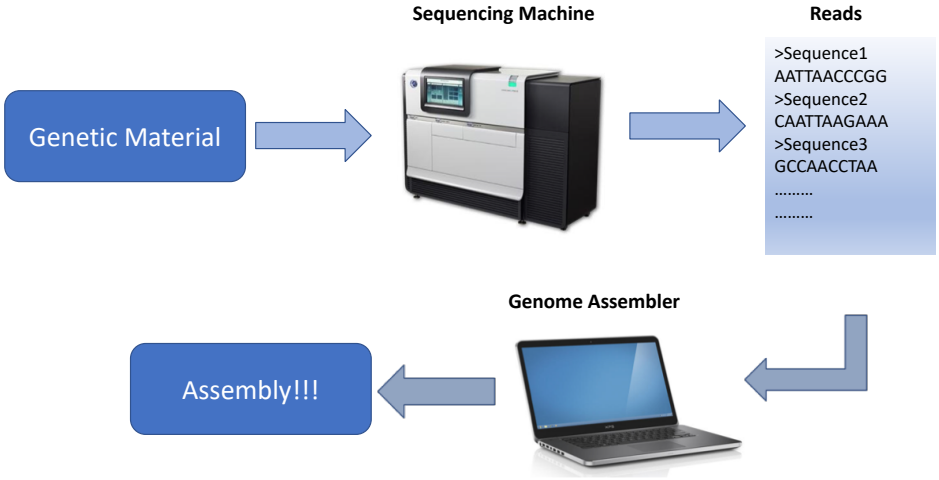


Figure 1.1: Overview of the genome assembly process. First, genetic material is sequenced, generating a collection of sequenced fragments (reads). These reads are processed by a computer program called an assembler which merges the reads based on their overlap to construct larger contiguous segments of the genome (contigs). Contigs are then oriented and ordered with respect to each other with a computer program called a scaffolder, relying on a variety of sources of linkage information. The scaffolds provide information about the long-range structure of the genome without specifying the actual DNA sequence within the gaps between contigs. The size of the gaps can also only be approximately estimated.

not yet completely solved. Also, newer sequencing technologies provide different kinds of information which facilitate the genome assembly process. Hence new algorithms need to be developed to accommodate such information.

In this chapter, we provide an overview of the genome assembly process and different paradigms used to solve the genome assembly problem. We then discuss metagenomics, where the goal is to understand the genomic sequence of multiple organisms simultaneously. We conclude the chapter with the outline of original contributions presented in the dissertation.

Technology	Read Length	Accuracy	Time per run	Bases per run
Single Molecule Real-Time Sequencing	5 kbp to 50 kbp	87% (Low)	30 minutes to 4 hours	5 10 Gb
Oxford Nanopore MinION Sequencing	5 kbp to 2000 kbp	70% to 90% (Low)	1 to 2 days	500 Mb
Ion Semiconductor	Up to 400 bp	98% (Medium)	2 hours	10Gb
Sequencing by synthesis (Illumina)	50-300bp	99.9% (High)	1 to 11 days	300 Gb
Sequencing by ligation	75 bp	99.9% (High)	1 to 2 weeks	3 Gb
Pyrosequencing (454)	700 bp	98% (Medium)	24 hours	400 Mb
Chain termination sequencing	400 to 900 bp	99.9% (High)	20 mins to 3 hours	50 - 100 Kb

Table 1.1: Overview of different sequencing technologies

1.1 Genome assembly

Genome assembly [4] (Figure 1.1) is the reconstruction of genomes from the smaller DNA segments, called reads, which are generated by a sequencing experiment. Various sequencing technologies have been developed in the past couple of decades that would help generate contiguous and accurate genome assemblies (see Table 1.1 for a summary of various sequencing technologies along with their properties). In many cases, reads are paired-end or mate-paired, which means that pairs of reads are sequenced from the ends of the same DNA fragment. The distance between the reads in each pair and their relative orientation are approximately known. This information is used to resolve ambiguities caused by repetitive sequences during assembly [5] as well as to order and orient the assembled contigs, the fragments of the genome that could be stitched together from the set of reads [6]. Below, we detail most commonly used approaches to solve the genome assembly problem.

Greedy: This is the most intuitive and straightforward method of assembly. In this method, individual reads are joined together into contigs in an iterative manner starting with the reads that overlap best and ending once no more reads or contigs can be merged.

This approach is simple to implement and useful in many practical settings and was used in several of the early genome assemblers such as TIGR [7], phrap [8], and VCAKE [9]. This simple greedy method, however, has some severe drawbacks. The choices made during the merging of reads/contigs are locally optimal and do not consider global relationships between reads. As a result, the approach can get stuck or can result in incorrect assemblies within repetitive sequences.

Overlap-Layout-Consensus: This three-step approach begins with a calculation of pairwise overlaps between all pairs of reads. The overlaps are computed with a variant of a dynamic programming-based alignment algorithm, making assembly possible even if the reads contain sequencing errors. This pairwise overlap information is used to construct an overlap graph where nodes are reads and edges denote overlaps between them. The layout stage consists of a simplification of the overlap graph to help identify a path that corresponds to the sequence of the genome. More precisely, a path through the overlap graph implies a 'layout' of the reads along the genome. In the consensus stage, a layout is used to construct a multiple alignment of the reads and to infer the likely sequence of the genome. Many assemblers used this assembly paradigm, including Celera Assembler [1], which was used to reconstruct the human genome, and Arachne assembler [10] used in many of the genome projects at the Broad Institute. The overlap-layout-consensus approach has also re-emerged recently as the primary paradigm used in assembling long reads with high error rates, such as those produced by the technologies from Pacific Biosciences and Oxford Nanopore.

De Bruijn graph: The de Bruijn graph assembly paradigm focuses on the relationship between substrings of fixed length k (k-mers) derived from the reads. The k-mers are

organized in a graph structure where the nodes correspond to the k-1 prefixes and suffixes of k-mers, connected by edges that represent the k-mers (Figure 1.2.) In this approach reads are not explicitly aligned with each other, rather their overlaps can be inferred from the fact that they share k-mers. With this graph, the assembly problem is reduced to finding a Eulerian path, a path through the graph that visits each edge once. The de Bruijn graph paradigm is affected by errors in the reads much more than the Overlap-Layout-Consensus paradigm as reads introduce false k-mers (false nodes and edges) in the graph. An assembler must eliminate these errors before identifying a Eulerian path in the graph. All practical de Bruijn assemblers include many heuristic strategies for eliminating errors from the reads and the graph. This paradigm has been widely used since the introduction of high throughput and relatively low-error sequencing technologies, in part because it is easy to implement and efficient even in the high depth of coverage settings. Some notable and widely used assemblers include: Velvet [11], SOAPdenovo2 [12], ALLPATHS [13], and SPADES [14].

Tradeoffs between different assembly methods: None of the methods described above is universally applicable; instead, each method has specific strengths and weaknesses depending on the characteristic of the data being assembled. The greedy method is easy to implement and is useful when the data contain no or only short repeats. The Overlap-Layout-Consensus approach is effective even at high error rates; however, its efficiency rapidly degrades with the depth of coverage and the number of sequences as it starts by computing all-versus-all overlaps. The de-Bruijn graph approach is computationally efficient even at high depths of coverage; however, it is affected by errors in the data and is, thus, most appropriate for relatively clean datasets.

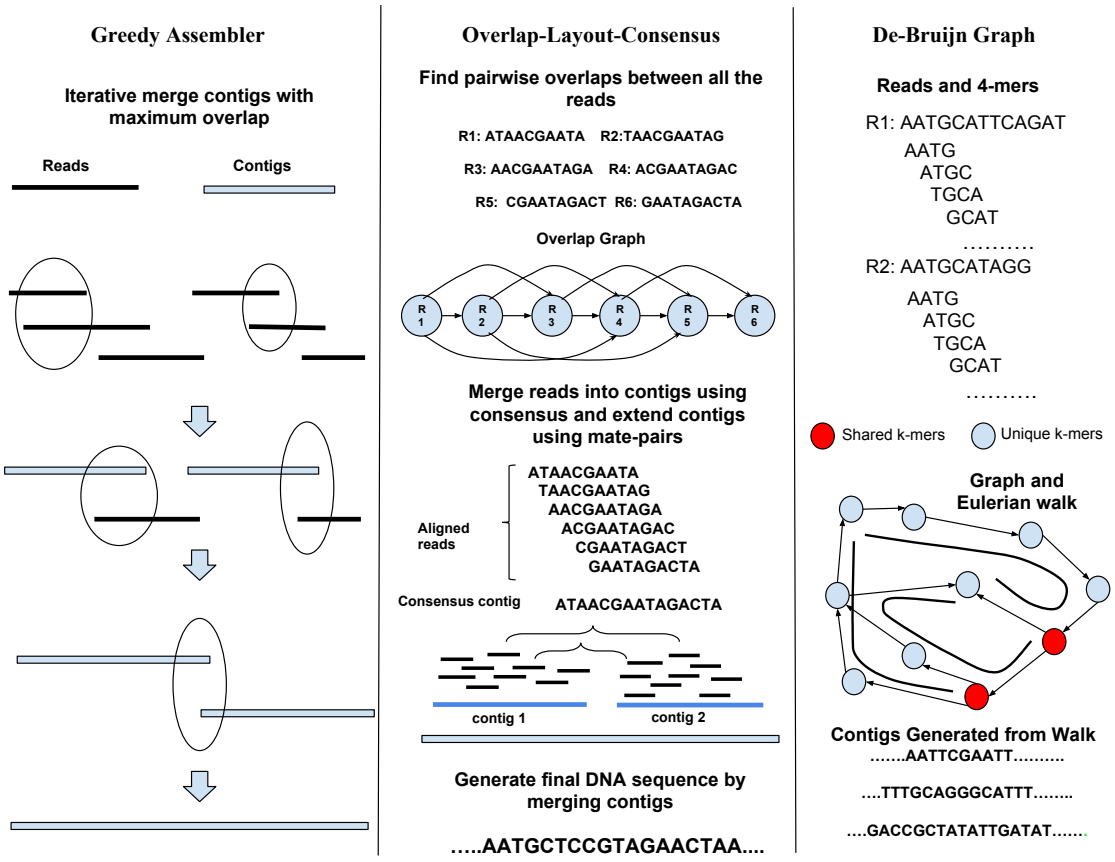


Figure 1.2: Overview of different *de novo* assembly paradigms. Schematic representation of the three main paradigms for genome assembly Greedy, Overlap-Layout-Consensus, and de Bruijn. Greedy assemblers merge reads with maximum overlaps are iteratively into contigs. In Overlap-Layout-Consensus approach, a graph is constructed by finding overlaps between all pairs of reads. This graph is further simplified, and contigs are constructed by finding branch-less paths in the graph and taking the consensus sequence of the overlapping reads implied by the corresponding paths. Contigs are further organized and extended using mate-pair information. In de Bruijn graph assemblers, reads are chopped into short overlapping segments (k-mers) which are organized in a de Bruijn graph structure based on their co-occurrence across reads. The graph is simplified to remove artifacts due to sequencing errors, and branch-less paths are reported as contigs.

1.2 Genome scaffolding

Despite the long length of sequences generated by third-generation sequencing technologies (tens of thousands of basepairs), the automated reconstruction of the entire genomes is difficult, mainly due to genomic repeats, ubiquitous features of both prokaryotic and eukaryotic genomes. These DNA segments that occur in two or more nearly-identical copies within genomes induce ambiguity in the reconstruction of a genome. The information contained in the reads alone cannot resolve this ambiguity. Furthermore, genomes also contain regions with unusual base-pair composition that are difficult to sequence. As a result, typical genome assemblies are highly fragmented, comprising tens to hundreds of thousands of contigs, without any knowledge of assignment of contigs to the chromosomes. Scientists recognized this fact from the early days of genomics, and have developed techniques that can generate information complementary to that contained in the reads. The assembly of the first living organism to be sequenced (*Haemophilus influenzae* [15]) relied on paired-read data that linked together relatively distant segments of the genome, allowing the assembled contigs to be ordered and oriented into a "scaffold" of the *H. influenzae* main chromosome. Depending on the information used to link contigs, the distance between adjacent contigs in the scaffold can also be estimated. Recently, new genomic technologies have been developed that can "bridge" across even longer repeats or other genomic regions that are difficult to sequence or assemble. These technologies are increasingly used to help improve genome assemblies by "scaffolding" together large segments of the genome. In Chapter 2, we review some of the technologies and algorithms used for genome scaffolding in detail.

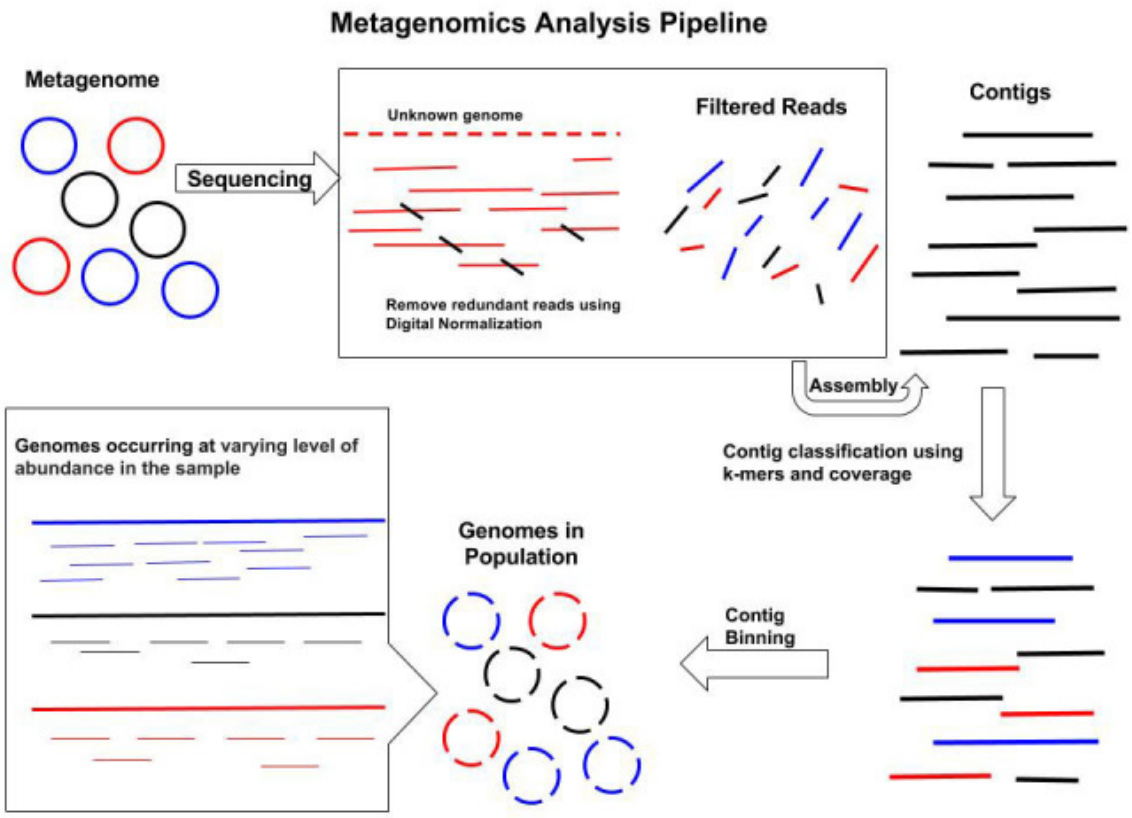


Figure 1.3: Metagenomic assembly pipeline: Multiple bacterial genomes within a community are represented as circles of different colors indicating multiple individuals from the same organism. Note the different levels of sequencing coverage for the individual organisms' genomes, due to the different abundance of the organisms in the original sample. The reads are then assembled into contigs, and they are classified using k-mers and coverage statistics. Contigs in each group are then binned to form draft genome sequences for organisms within the population.

1.3 Metagenomics

Metagenomics is a fairly new research field focused on the analysis of sequencing data derived from mixtures of organisms. It also provides a means to study unculturable organisms that are otherwise difficult or impossible to analyze. Metagenomic data consists of a mixture of DNA from different organisms and may comprise viral, bacterial, or eukaryotic organisms. The different organisms present in a mixture may have widely dif-

ferent levels of abundance, as well as different levels of relatedness with each other. These characteristics complicate the assembly process. As we described above, one of the main challenges to the assembly of single organisms is due to repetitive DNA segments within an organism's genome. For a single organism, assuming a uniform sequencing process, such repeats can be detected simply as anomalies in the depth of coverage (a two-copy repeat would contain twice as many reads as expected). Due to the uneven (and unknown) representation of the different organisms within a metagenomic mixture, simple coverage statistics (such as flagging contigs with coverage higher than twice the average assembly coverage or A-stat [16]) can no longer be used to detect the repeats. The confounding effect of repeats on the assembly process is further exacerbated by the fact that unrelated genomes may contain nearly-identical DNA (inter-genomic repeats) representing, for example, mobile genetic elements. At the other extreme, the multiple individuals from the same species may harbor small genetic differences (strain variants). The decision of whether such differences can be ignored when reconstructing the corresponding genome, or whether it is appropriate to reconstruct individual-specific genomes is not only computationally difficult but also ill-defined from a biological point of view. Furthermore, distinguishing true biological differences from sequencing errors becomes nearly impossible in a metagenomic setting. Depending on the nature of the study, metagenomic analysis falls into two different categories as follows.

1.3.1 Marker gene analysis

Certain genes in bacterial and fungal genomes are conserved across different species and typically contain a highly variable region which is flanked by highly conserved regions. These are called marker genes. Some examples of marker genes are 16s rRNA and 18s rRNA in bacteria and internal transcribed spacer (ITS) in fungi. These highly conserved flanking regions in marker genes provide binding sites for PCR primers. Marker gene amplification and sequencing are fast and cost-effective methods to obtain a low-resolution composition of the microbial community. This approach works well for samples with host DNA contamination. The typical 16s rRNA analysis pipeline starts with using primers designed to amplify and sequence one or a few hypervariable regions of the 16S rRNA gene. These sequences are then binned into groups called "Operational Taxonomic Units" (OTUs) based upon sequence similarity [17, 18]. From each OTU cluster, a single sequence is selected as a representative sequence. The representative sequence is then annotated using a 16s rRNA gene classification method [19], and all sequences within the OTU inherit that same annotation. Several pipelines [20, 21] have been developed to perform 16S rRNA gene analysis from end to end. Marker gene sequencing usually correlates well with genomic content [22, 23]. Although this method provides a quick, simple, and inexpensive way of preparing and analyzing samples [20, 24], there are certain drawbacks. Marker gene analysis is typically subjected to amplification biases [25]. Since a part of the entire genome is sequenced (typically one or a few genes), the resolution of the composition of a community is limited to a genus level at best. One more drawback of 16s rRNA gene analysis is that some families of bacteria possess a

very similar 16s rRNA variable region, despite being genotypically and phenotypically divergent. Because of this, the functional profiling of such bacterial communities may not be accurate by using only marker gene data [26, 27].

1.3.2 Whole metagenome sequencing

Whole metagenome sequencing aims to sequence all genomes within a sample and hence has a potential to provide better genomic information and taxonomic resolution than marker gene sequencing. Contrary to marker gene analysis, whole metagenome sequencing can profile functional characteristics of an entire community at the gene level [28]. However, it is relatively expensive to prepare, sequence, and analyze a whole metagenome sample. Whole metagenome sequencing also captures all DNA in the sample, including viral and eukaryotic DNA. By sequencing samples at adequate sequencing depth, taxonomic resolution up to species or strain level can be achieved [29] using just the reads. In some cases, reads are assembled to generate longer contig sequences which can simplify bioinformatics analysis compared to unassembled short reads. These contigs are then binned into separate putative genomes based on metrics such as coverage statistics, GC content, and k-mers composition [30, 31]. Figure 1.3 shows a typical pipeline to obtain binned metagenomes starting from sequenced reads. With a fragmented but high-quality metagenome assembly, the genetic repertoire of a microbial community can be identified using genome characterization tools. These include a gene identification step [32, 33], followed by functional annotation pipelines [34, 35] commonly used for characterizing genome assemblies. Some of the largest whole metagenome sequencing

projects [36] have used metagenomic assemblies to construct the microbial gene catalog for human [37], mouse [38], and dog [39] gut metagenomes. Thus, whole metagenome sequencing and analyses provide valuable insights into the microbial organisms that biologists are unable to culture in a lab due to lack of knowledge about the required growth conditions.

1.4 Contributions

In Chapter 2, we review both historical and contemporary sequencing technologies and scaffolding algorithms that are being used for assembling large genomes. We also survey some of the recent sequencing projects which used these data to generate accurate and contiguous assemblies of different species.

In Chapter 3, we present a novel method, SALSA2, that utilizes the chromosomal contact information obtained from Hi-C data along with a genome assembly overlap graph to improve the continuity and accuracy of the genome assembly. Placing contigs of large eukaryotic genomes into chromosomes when in most cases the number of chromosomes is often not known, is an essential step in genome assembly, and we have provided a software tool which can be easily used to perform this task.

In Chapter 4, we explore different types of Hi-C library preparation which can be useful in multiple steps of genome finishing. Traditionally, Hi-C data has only been used for placing contigs along chromosomes. We explore the possibility of devising a Hi-C library that can be used for scaffolding, base level polishing, and haplotype phasing of genomes. We demonstrate the utility of such library on the human and chicken genomes.

In Chapter 5, we explore one of the critical issues in metagenome scaffolding: detecting repeats. The nature of repeats in metagenomes differs from that of single genomes because of non-uniform sequencing coverage and the presence of closely related species in the sample. We devise a method to utilize features derived from scaffold graph topology to identify repetitive sequences that complicate the scaffolding process.

In Chapter 6, we propose a new scaffolding pipeline, MetaCarvel, that can scaffold large and complex metagenomic datasets correctly. We have devised a variant detection method that preserves the variation between species in a sample when generating scaffolds. We show the accuracy of our approach on three different kinds of variants and confirm that the scaffolds generated by MetaCarvel agree the most with underlying sequencing data.

In Chapter 7, we develop a novel method based on Four Russians' speedup for dynamic programming to cluster sequences from 16s rRNA metagenomic studies. We exploit the similarity between the sequences to develop a trie-like data structure to facilitate the fast similarity search. We show that our method runs exponentially faster than the existing method while clustering millions of sequences from a single study at high similarity threshold to generate species-level OTUs.

Chapter 2: Related work

In this chapter, we survey recent advances in this field, placed in the historical context of the technologies and algorithms that have been used for scaffolding throughout the entire genomic revolution. This chapter would provide the background for the later part of the dissertation.

2.1 Source of information for genome scaffolding

Any genomic information that hints at the relative location of genomic segments along a genome can be used to drive the scaffolding process. In most cases, the information used derives from genomic technologies specifically designed to interrogate the structure of genomes, though indirect inferences based on evolutionary arguments have also been used effectively in genome scaffolding. Figure 2.1 shows the extent to which different sequencing technologies can provide the linkage information for scaffolding. This linkage information can span anywhere from several hundred to thousands of basepairs (Illumina and Pacific Biosciences) to hundreds of thousands of basepairs (Linked Reads and Optical Maps) to millions of basepairs (Chicago and Hi-C). We organize these approaches into broad categories defined by their key characteristics.

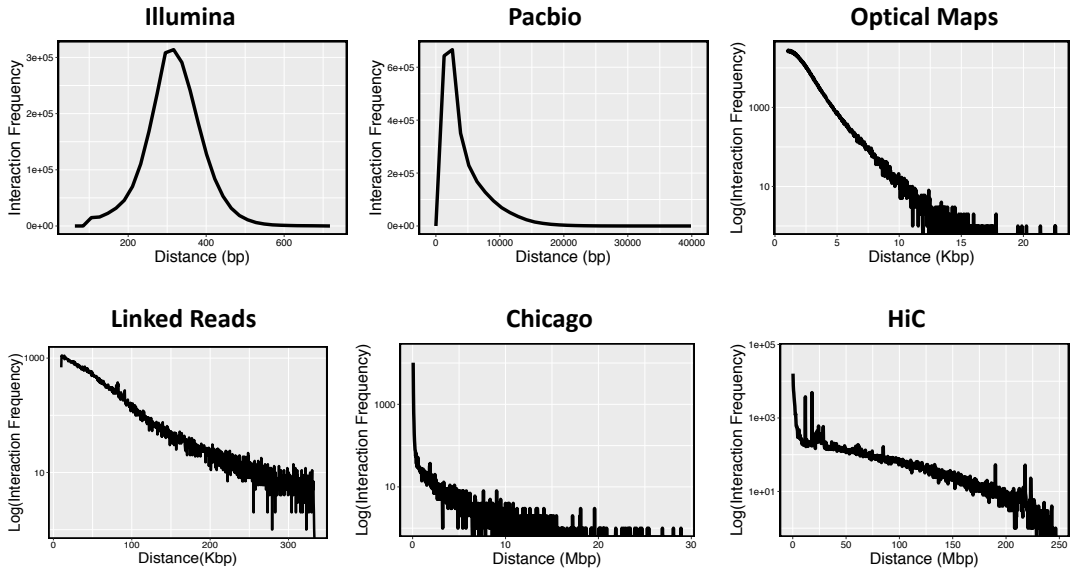


Figure 2.1: The genomic separation for pairwise linkage information provided by different sequencing technologies for NA12878 human genome sequencing data when mapped to the GRCh38 human reference genome. Illumina, Linked Reads, Chicago and Hi-C reads were aligned using BWA-MEM program. Optical maps were aligned using RefAligner program. Pacbio long reads were aligned using BLASR program. The distributions for Optical maps, Linked Reads, Chicago and HiC are in log scale to accommodate the large variance in the linking distance.

2.1.1 Physical mapping technologies

Physical mapping attempts to estimate the location of specific loci along genomic chromosomes. The loci can be short DNA segments that are unique within the genome, as in the case of Sequence-Tagged Sites (STS) [40], or the recognition sequence of a restriction enzyme, as in the case of restriction mapping and optical mapping. Fluorescent in situ hybridization (FISH) [41] locates the position of markers by hybridizing fluorescently labeled probes to intact chromosomes. Radiation Hybrid mapping (RH mapping) [42] uses the random breakage of chromosomes by X-rays to determine the distance between DNA markers, along with their order on the chromosome. Restriction maps [43] measure the

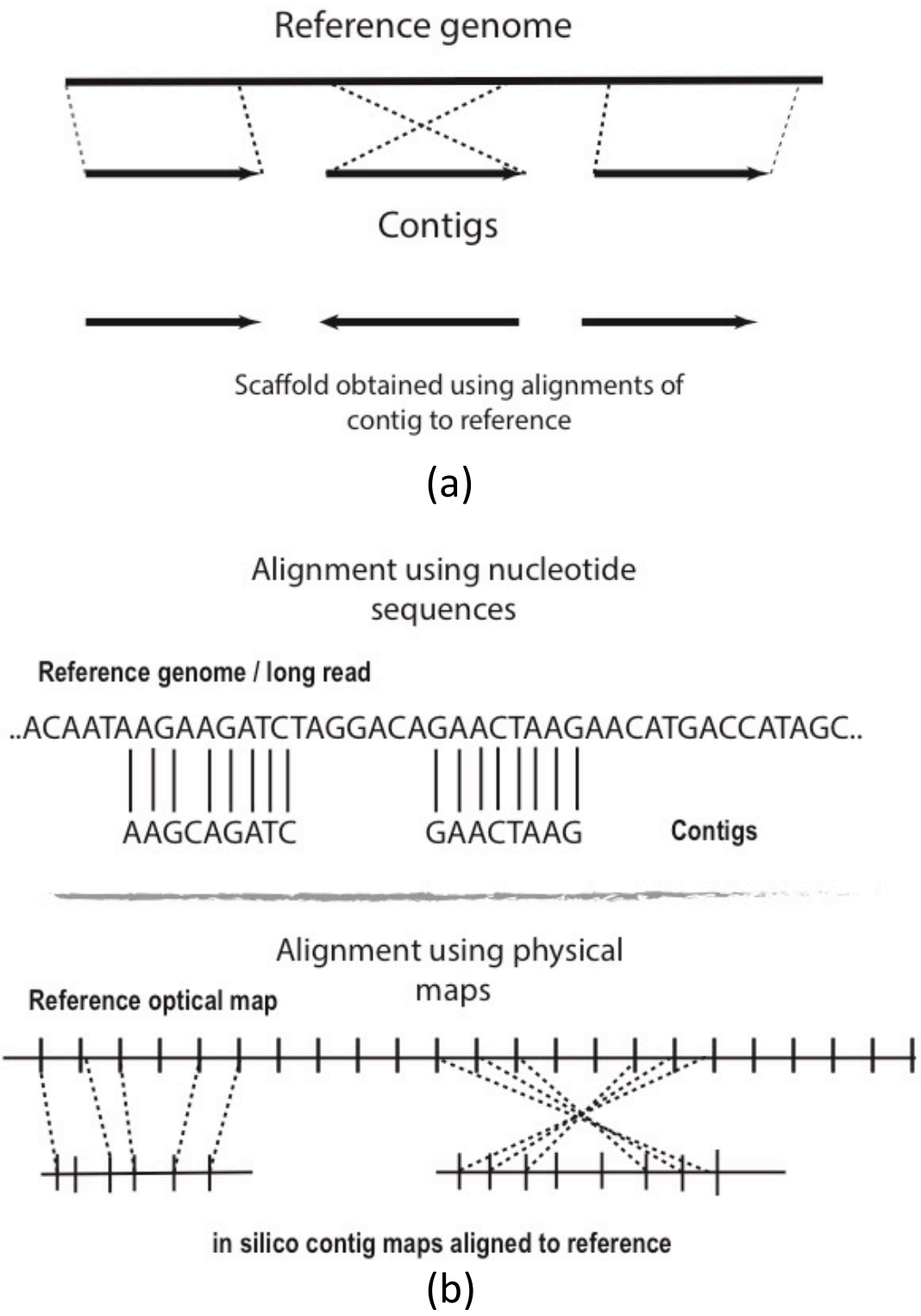


Figure 2.2: Mapping-based scaffolding approaches. Contigs are aligned to a reference genome to infer their relative order and orientation. The alignment can be based on the actual nucleotide sequence (a) when the reference is a (partially) sequenced genome, or a physical map (b) when only a sparse map of the reference genome is available. In the latter case, an in silico map is constructed from the sequenced contigs before alignment.

distances between endonuclease recognition sites within a DNA molecule. The original use of restriction enzymes to map chromosomes resulted in unordered information - simply the list of sizes of the restriction fragments generated from the molecule. Optical maps are an enhancement of restriction mapping which provides the fragment order in addition to their size [44]. Optical map experiments start with high molecular weight DNA molecules which are immobilized on a surface or passed through a nanochannel [45] (the latter technology is called nanocoding). The DNA is digested/nicked with one or more restriction enzymes and colored with a fluorescent dye. The distance between cuts is detected by imaging the DNA and integrating the fluorescence intensity. The resulting data is an ordered series of fragment sizes estimated from the machine imaging of the distances between restriction cuts.

The use of physical maps in ordering genomic contigs along a chromosome has been extensively studied since the early days of genomics [46–48]. Broadly, experimentally derived maps are compared to theoretical maps generated from the sequenced contigs in order to determine the location of the contigs along a chromosome (Figure 2.2.) This process is easiest when the landmarks being compared are distinguishable from each other (as is the case for STS and radiation hybrid maps) and substantially more complex and error-prone for restriction maps where all the landmarks are identical in sequence [49].

The experimental maps themselves are often the result of assembling a collection of DNA fragments (clones) that have been mapped separately, leading to similar analytical challenges as those encountered in genome assembly [50]. In the case of unordered restriction maps, the assembly process is guided by the probability that two clones overlap, the probability that is computed by taking into account the number of restriction frag-

ments shared by the clones [51]. The pairwise overlap probabilities are used to assemble the clones into a chromosome-wide structure using a heuristic assembly algorithm (FPC) which also allows for manual intervention to inspect and correct the resulting layout. Ordered restriction maps, as generated by optical or nanocoding mapping, can be aligned using variants of dynamic programming alignment algorithms [52, 53]. In SOMA [54] fragment sizing errors are penalized through a chi-squared scoring function, and a variant of a scheduling algorithm is used to determine the layout of contigs with ambiguous mappings. The running time of the alignment algorithm used in SOMA scales with the fourth power of the number of fragments, making the approach impractical for large genomes. Two recent approaches address this limitation. TWIN [55] relies on an extension of the FM-Index [56] to speed up the alignment process, while Maligner [57] indexes the reference map by simulating the effect of common mapping errors such as false cuts or missed restriction sites.

2.1.2 Paired-read technologies

Some sequencing technologies can provide additional information about the relative placement of reads along the genome being sequenced. The most widely used information links together pairs of reads whose distance and relative orientation along a chromosome can be estimated through the sequencing experiment. Most commonly, this information is derived by carefully controlling DNA shearing before sequencing in order to obtain fragments of uniform sizes, and by tracking the link between DNA sequences "read" from the same fragment. Multiple protocols have been developed over the years to

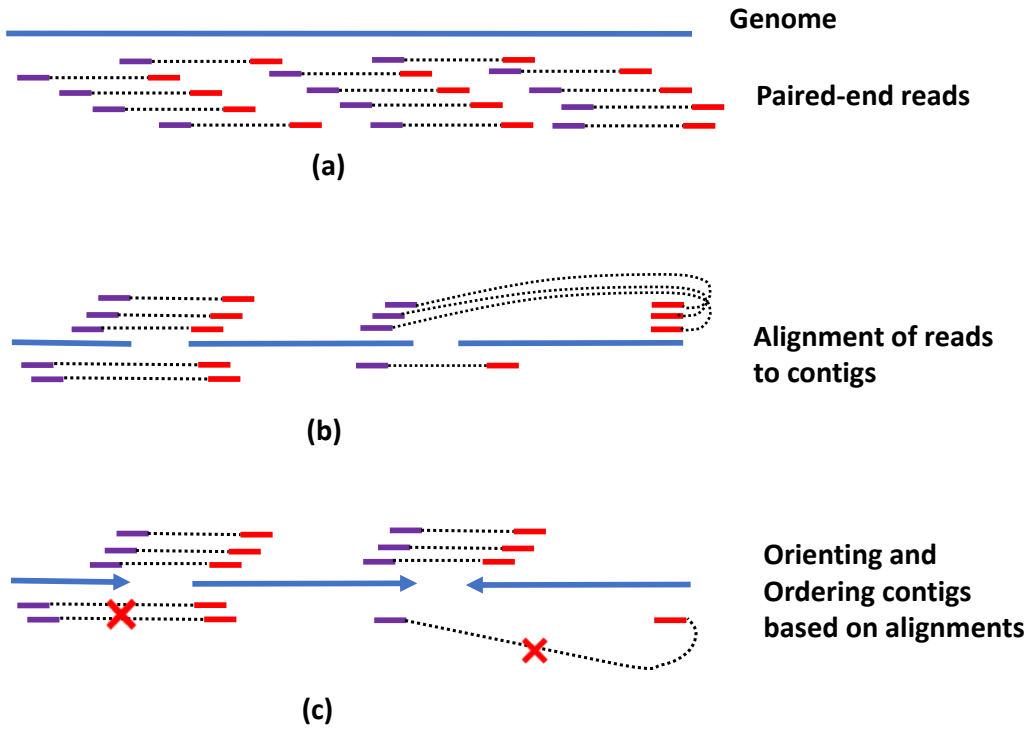


Figure 2.3: Use of pairwise linkage information for scaffolding. (a) Paired-end reads are sequenced from the genome. Depending on the technology, the approximate distance and/or relative orientation of the paired reads may not be known. (b) The reads are aligned to contigs. Reads with their ends aligned to two different contigs provide linkage information useful for scaffolding. (c) Linkage information is used to orient and order the contigs into scaffolds. Usually, not all constraints can be preserved, and algorithms attempt to minimize inconsistencies (marked with X).

generate read pairing information, and different names are commonly used to reflect the experimental source: paired-end reads (pairings natively generated by Illumina sequencing instruments, usually short-range 300-500bp), mate-pair or jumping libraries (pairing information derived with the help of additional experimental assays, usually spanning thousands to tens of thousands of base-pairs). Throughout this dissertation, we will use these terms interchangeably as the information being generated is the same - pairs of reads with an approximately known relative distance and orientation. The information provided by mate pairs can be used to link the contigs and produce scaffolds [58]. Mate pairs can also be used to resolve repeats in the genome assembly process. The algorithms for using mate-pair data in scaffolding genomes all follow a similar workflow. First, mate pairs whose ends map to different contigs are used to link together the corresponding contigs. Second, the pairwise linkage information is used to orient and order contigs with respect to each other. Third, the size of the gap between adjacent contigs is estimated from the experimentally determined size of the mate-pairs, and a linear layout of the contigs along a scaffold is generated (Figure 2.3). Since contig orientation and ordering are NP-Hard problems, scaffolders implement different greedy heuristics. Scaffolders such as MIP [59], SOPRA [60] and SCARPA [61] use integer programming to find the optimal orientation and ordering of contigs. Bambus [62] and SSPACE [63] hierarchically use multiple libraries to perform scaffolding, starting from libraries with smaller insert size (which are more accurate and yield a simpler problem), and progressively expanding scaffolds using libraries with larger insert sizes. OPERA-LG [64] uses a branch and bound search to determine the relative placement of contigs along the chromosome. The authors show that the size of the search space is bounded by the ratio between the library and

contig size, implying that the branch and bound heuristic is efficient for the data typically encountered in practical applications despite a theoretically exponential complexity.

Paired-end reads have also been extensively used to resolve repeats. The orientation and distance constraints imposed by paired reads limit the number of possible traversals of the graph through a repeat region and can link together the unique genomic regions surrounding each instance of a repeat. Assemblers such as Velvet [11], Abyss [65] and IDBA-UD [66] use paired-end information to guide the walk through the assembly graph. SPAdes [14] and metaSPAdes [67] assemblers use the ratio of expected to the observed number of mate pairs connecting two nodes [68] in the de Bruijn graph to check if the path traverses through a repetitive region. Wetzel et al. [69] explored the extent to which mate pairs can be used to resolve repetitive regions in prokaryotic genomes and showed that mate pair libraries are most effective if tuned to the structure of the assembly graph.

2.1.3 Chromosomal contact data

Several techniques have been developed recently to study the three-dimensional structure of chromosomes inside a cell [70]. These techniques are collectively referred to as chromosomal conformation capture (C3). The two most widely used variants rely on DNA sequencing and generate pairwise linking information between reads that originate from genomic regions that are physically adjacent in a cell. Unlike the standard mate-pair data, the distance and the relative orientation between the paired reads is not known a priori.

The two most commonly used protocols for capturing chromosome conformation

are Hi-C [70] and Chicago [71]. In the Hi-C protocol, DNA in the cell nucleus is crosslinked and cut with a restriction enzyme. This process generates fragments of DNA which are distally located but physically associated with each other. The sticky ends of these fragments are biotinylated and then ligated to each other to form a chimeric circle. These biotinylated circles are sheared and processed into sequencing libraries in which individual templates are chimeras of the physically associated DNA molecules. The Chicago protocol from Dovetail Genomics starts not with cells but with purified DNA so that any biologically associated chromosomal interactions are eliminated. Artificial nucleosomes with random specificity are then used to condense the DNA into chromatin, which is then processed through the standard Hi-C protocol. The result is a collection of fragments which is enriched for sets of paired reads that capture long-range interactions between segments of DNA that were in contact within the artificial chromatin.

Since Hi-C and Chicago protocols do not provide good estimates of the distance between the paired reads, the data can only be used to estimate the relative order and orientation of contigs accurately. The scaffolding process starts by filtering the data to eliminate artifacts such as reads aligning to multiple locations or chimeric reads derived from the ligation junctions. Several tools have been developed for this purpose, including HiCUP [72], HiCPro [73], Juicer [74], Juicebox [75] and HiFive [76] . These tools align reads to the assembly using standard alignment programs [77–79], and filter the alignments to remove experimental artifacts, yielding the 'true' alignments which imply the contact information. The number of paired reads linking two genomic regions (contact frequency) correlates with the one-dimensional distance between the corresponding regions, thereby yielding an estimate of the relative placement of these segments within

a genome. Furthermore, the contact frequency is much higher within a chromosome than across chromosomes, making it possible to infer chromosome structure directly from the genome assembly. Most of the algorithms developed to use Hi-C data for scaffolding use these properties to group contigs into chromosome-specific bins, then orient and order the contigs within each chromosome by maximizing the concordance with the experimentally-derived contact frequencies.

DNATri [80] and LACHESIS [81] were the earliest methods developed to use Hi-C datasets for scaffolding. DNATri relies on a limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimization algorithm [82] to identify the placement of contigs that best matches the contact frequencies derived from the Hi-C data. LACHESIS first clusters the contigs into chromosome groups using hierarchical clustering, matching a user-specified number of chromosomes. Then, it orders and orients contigs in each chromosome group/cluster separately by formulating the problem as identifying the "trunk" of a minimum spanning tree of the graph that encodes the Hi-C links between contigs. GRAAL [83] models the Hi-C data by distinguishing between cis- contacts (occurring within the same molecule) and trans- contacts (occurring across molecules). The contact frequency for the former is distance dependent, while the latter are drawn from a uniform probability distribution. The contigs are ordered and oriented to maximize the fit with this modeled data using a Metropolis optimization algorithm [84]. SALSA [85] relies on Hi-C data to correct misassemblies in the input contigs and then orients and orders the contigs using a maximal matching algorithm. 3D-DNA [86] also corrects the errors in the input assembly and then iteratively orients and orders unitigs into a single megascaffold. This megascaffold is then broken into a user-specified number of chromosomes, identifying

chromosomal ends based the on Hi-C contact map. Putnam et al. [71] proposed a method called Hi-Rise that was specifically designed for handling Chicago libraries (based on artificial chromatin). They rely on a likelihood function that matches the characteristics of these data and use dynamic programming to identify a layout of contigs that maximizes the fit with the experimental data.

2.1.4 Subcloning

Subcloning involves breaking up the genome into large fragments that are then sequenced separately, retaining the connection between the sequencing reads generated from the same subclone. The assembly process can then be run for each subclone separately, and the resulting assemblies merged to reconstruct the full genome sequence. Initially, subcloning relied on Bacterial Artificial Chromosome (BAC) cloning, yielding fragments in the range of 100 kbp in length. The two ends of each BAC clone were sequenced first in order to construct a clone map [87] representing the relative relationship between individual clones along the genome. From this information, a minimal tiling path was identified in order to decide which clones would be fully sequenced. Researchers used this strategy in the early days of genomics, most notably during the public effort to sequence the human genome [2].

The BAC cloning process involved growing the fragments in an *Escherichia coli* culture, and it was, thus, highly expensive and labor intensive. Recently, several groups are developing new technologies that perform the subcloning process *in vitro*. The technology from 10x Genomics partitions large DNA fragments into droplets, and the DNA

is sheared, and sequencing libraries are constructed within the droplets. The DNA within each droplet is tagged with a droplet-specific barcode, and these barcoded DNA molecules then undergo sequencing and a post-processing algorithm parses the barcodes to group the reads originating from a same large DNA fragment [88].

Illumina commercializes similar technology under the name TruSeq Synthetic Long-Read (TSLR) [89]. This technology fragments DNA into large segments of about 10 kb in size and distributes into pools such that each pool contains a relatively small number of fragments (about 200-300). Each pool is processed separately and barcoded with a unique barcode before sequencing.

When the original fragments have been sequenced deeply enough (which is usually the case for the TSLR technology), the pooled reads can be assembled in order to create complete reconstructions of the individual fragments, effectively generating long and highly accurate sequencing reads. Several approaches have also been developed that rely on the unassembled pooled reads to guide the scaffolding process, approaches that can be effective even at low depths of sequencing coverage. Fragscaff [90], a method which was initially designed for contiguity-preserving transposition sequencing data creates links between the ends of contigs which are represented in the set of reads from the same pool. Within the resulting graph, Fragscaff then identifies a minimum spanning tree by treating edge weights as the number of pools shared between contigs and the longest path within this tree is selected as the scaffold backbone. In Supernova, Weisenfeld et al. [91] use the mate-pair information to span short gaps and construct an initial set of scaffolds, after which the pool-specific barcodes are used to construct an adjacency graph where the nodes are the initial set of scaffolds and edges denote the number of pool-specific bar-

codes shared between scaffolds. In this graph, it then finds linear paths consistent with the links provided by barcodes. For each linear path, Supernova lists all the alternative linear paths which are sharing a large number of barcodes. By scoring the orientation and orderings of all these paths, the path with the highest score is chosen to scaffold the contigs.

2.1.5 Conservation of synteny

Synteny refers to the co-localization of genes or genomic loci along a chromosome. In many cases, while the DNA sequence itself may diverge significantly during evolution, related organisms often preserve synteny and gene order. The conservation of synteny can, thus, be used to help order contigs along a chromosome by inferring their placement based on the location within a related genome of the orthologs of the genes found in the contigs.

Synteny-based methods first map contigs onto the reference genomes using a whole genome aligner [92, 93]. The orientation and ordering of contigs are then inferred from the alignment data (Figure 2.2). Methods such as OSLay [94], ABACAS [95], Mauve Aligner [96], fillScaffolds [97], r2cat [98] and CAR [99] use only one reference genome for scaffolding draft assemblies. They are primarily based on mapping the assembly to a complete or incomplete reference genome and attempt to identify the ordering and orientation of contigs that is most consistent with the reference genome. The main challenges involve reconciling true differences with the reference genome as well as handling incomplete reference genomes. Such approaches may lead to mistakes if the reference genome

is re-arranged with respect to the genome being assembled, or if the two genomes are too distant in phylogenetic terms. MeDuSa [100] and Ragout [101] use multiple reference genomes along with the phylogenetic tree of these genomes as a reference to scaffold contigs. MeDuSa models the problem of using multiple reference genomes for scaffolding as an instance of maximum weight path cover problem [102] which is known to be NP-hard, and propose a greedy heuristic to find solution close to the optimal solution. Ragout represents the target and reference genomes as a multi-colored breakpoint graph [103] with nodes representing the conserved synteny blocks and edges representing the adjacency of these blocks. In this graph, Ragout finds the missing adjacencies by solving a half-breakpoint state parsimony problem on the given phylogenetic tree [104] and then orients and orders synteny blocks to reconstruct the target genome. Multi-CAR [105] starts by processing each reference genome separately using CAR. Multi-CAR then reconciles the different contig orderings by constructing a graph where nodes are contigs and edges are the adjacencies given by different reference genomes. A maximal weight perfect matching within this graph defines the final set of scaffolds.

2.1.6 Scaffolding based on long reads

Sequencing technologies that generate long sequencing reads, such as Pacific Bioscience [106] and Oxford Nanopore [107], as well as the above-mentioned in vitro subcloning approaches, have tremendously matured over the past few years and have contributed dramatically to genome assembly. To achieve high-quality assemblies with only long read data, however, the genome needs to be sequenced at considerably high cover-

age, incurring significant costs. A more cost-effective strategy involves supplementing a short-read assembly with a relatively low coverage long read data. SSPACE-LongRead [108] was one of the earliest methods able to leverage long reads for scaffolding (Pacific Bioscience's RS, in this case). It aligns long reads to pre-assembled contigs using BLASR [109] in a local alignment mode and uses the alignment coordinates to infer orientation, ordering and the distance between contigs. LINKS [110] is an alignment-free approach designed to use data generated by Oxford Nanopore's MinION sequencer. It extracts pairs of k-mers from long reads which are separated by a pre-defined distance, then treats these as if they were mate-pairs and uses a traditional scaffolding algorithm to order and orient the genomic contigs.

2.2 Gap filling

The scaffolds obtained with pairwise linkage information often contain gaps separating adjacent contigs. Mate pairs which have one end aligned to a contig and the other positioned within the gap can be used to infer the unassembled sequence within the gap. GapFiller [111] and SOAPdenovo GapCloser [12] are standalone methods which implement this approach. Sealer [112] uses a succinct representation of de Bruijn graphs using Bloom filters [113] to reduce the memory requirement, thereby enabling gap filling in large draft genomes. Assemblers such as ABySS [65], ALLPATHS-LG [13] and EULER [114] implement built-in gap filling modules which use mate-pair data in a similar way. The PBJelly [115] method uses flanking read sequences obtained from the mapping of PacBio long reads to assemblies to fill the gaps. GMCloser [116] uses both paired-end

reads and long reads to close the gaps. It first aligns contigs to long reads and extracts all end-to-end alignments between them. It then finds a significant end to end alignments using a likelihood ratio test based on the alignment of paired-end reads to both contigs and long reads.

2.3 Hybrid scaffolding

The data used for scaffolding can contain errors and other artifacts. For mate pairs, incorrect insert size estimates can lead to ordering and gap estimation errors in the scaffolds [117]. Hi-C data cannot provide accurate orientation information at small genomic distances, yielding small inversions within the scaffolds [118]. Optical mapping data have relatively low resolution and contain many errors such as incorrect estimates of fragment sizes and missed or spurious restriction cuts [119]. A combination of multiple complementary technologies can be used to reduce the impact of errors. Pendleton et al. [120] used Pacific Biosciences long read assemblies to assemble the NA12878 human genome and scaffolded these assemblies with BioNano Genomics optical map data. Mostoyov et al. [88] used Illumina short reads to perform contig assembly and used 10x Genomics linked-read data along with optical map data to get high quality, haplotype-phased *de novo* assembly of the human genome. Bickhart et al. [118] used Pacific Biosciences' PacBio RSII long read data to assemble the domestic goat genome. After the assembly, they used BioNano Genomics optical mapping data to scaffold the contig assembly and correct the assembly errors. These optical map scaffolds are further scaffolded with Hi-C data to group scaffolds into chromosomes. Seo et al. [121] used Pacific Biosciences RSII long

read data along with BioNano Genomics optical maps and 10x Genomics linked read data to generate a high quality phased assembly of the genome of an individual. Multiple sequencing technologies have also been used to assemble more complicated plant genomes. Mascher et al. [122] used a combination of optical maps and Hi-C data to get a chromosome level assembly of the barley genome. Du et al. [123] used a combination of long insert size libraries and optical maps to assemble the indica rice genome. Jiao et al. [124] combined optical map and Hi-C data to produce assemblies of relatives of *Arabidopsis thaliana*.

2.4 Haplotype phasing

Humans are diploid organisms and have two copies of each chromosome (except the sex chromosomes). The two "haplotypes" represent the complete information on DNA variation in an individual. Reconstructing individual haplotypes is extremely for understanding human genetic variation, linking variants to disease, and reconstructing human population history [125–128]. Haplotypes for an individual genome at known heterozygous variants can be directly reconstructed by aligning sequenced reads to the reference genome. Reads that are long enough to cover multiple heterozygous variants provide partial haplotype information. Using overlaps between such haplotype-informative reads, long-range haplotypes can be inferred. Different computational methods have been developed to use new types of sequencing data such as PacBio, 10x Genomics linked reads, and Hi-C reads to generated chromosome-sized phased haplotypes of large genomes [129–132].

2.5 Conclusion

In this chapter, we reviewed some of the historical and contemporary advances in the sequencing technologies and their impact on the development of computational methods. These emerging long read sequencing and mapping technologies, coupled with new algorithms have started to produce reference genomes of unprecedented quality. The high-quality genome assemblies have been achieved with the longest possible reads, complemented by the longest possible mapping information provided by 10X Genomics linked reads, Hi-C, or BioNano Genomics data for scaffolding. As these technologies mature, both regarding cost and quality, it is likely that many projects will begin with fully assembled genomes rather than just the variants obtained by aligning reads to the reference genomes. Hence, our contributions in the area of long-range scaffolding with Hi-C data are useful for a broader research community.

Repetitive sequences remain a challenge even for single genomes and their effect in metagenomic data is further amplified by the presence of cross-organismal repeats and uneven levels of representation of organisms within a sample. New sequencing technologies such as PacBio and Oxford Nanopore that provide long but error-prone reads can overcome some of the challenges posed by repeats; however, these approaches are still too expensive to be applied in a metagenomic data. Algorithms for long read assembly will need further development before it can be used effectively in a metagenomic setting. Because of this, short paired-end reads remain the primary source of data for metagenomic assemblies. Our contributions in using paired-end read data for repeat detection, variant detection, and variation-aware scaffolding address some of the challenges specific

to metagenomic datasets. Furthermore, targeted studies based on the 16S rRNA gene have already generated a wealth of data about microbial communities, primarily restricted to information about the taxonomic origin of organisms. Our contributions in 16S rRNA data clustering provide a computationally efficient way to analyze these datasets.

Chapter 3: Scaffolding large eukaryotic genomes with HiC data

3.1 Introduction

Genome assembly is the process of reconstructing a complete genome sequence from significantly shorter sequencing reads. Most genome projects rely on whole genome shotgun sequencing which yields an oversampling of each genomic locus. Reads originating from the same locus are identified using assembly software, which can use these overlaps to reconstruct the genome sequence [133, 134]. Most approaches are based on either a de Bruijn [114] or a string graph [135] formulation. Repetitive sequences exceeding the sequencing read length [136] introduce ambiguity and prevent complete reconstruction. Unambiguous reconstructions of the sequence are output as "unitigs" (or often "contigs".) Ambiguous reconstructions are output as edges linking unitigs. Scaffolding utilizes long-range linking information such as BAC or fosmid clones [81, 137], optical maps [43, 138, 139], linked reads [91, 140, 141], or chromosomal conformation capture [142] to order and orient unitigs. If the linking information spans large distances on the chromosome, the resulting scaffolds can span entire chromosomes or chromosome arms.

Hi-C is a sequencing-based assay originally designed to interrogate the 3D structure of the genome inside a cell nucleus by measuring the contact frequency between all pairs

of loci in the genome [70]. The contact frequency between a pair of loci strongly correlates with the one-dimensional distance between them. In the data generated by the Hi-C protocol, the intrachromosomal contact probability is on average much higher than the interchromosomal contact probability. Regions separated by several hundred megabases on the same chromosome are more likely to interact than regions on different chromosomes, though it is important to note that the interaction probability rapidly decays with increasing genomic distance. The main advantage of Hi-C over previous methods is the ability to capture interactions over much larger genomic distances thereby producing scaffolds which can span a whole arm of the chromosome. Hi-C data can provide linkage information across a variety of length scales, spanning tens of megabases. As a result, Hi-C data can be used for genome scaffolding. Shortly after its introduction, Hi-C was used to generate chromosome-scale scaffolds [80, 83, 86, 118, 143].

LACHESIS [143] is an early method for Hi-C scaffolding. LACHESIS uses the Hi-C reads alignments to contigs to cluster contigs into one cluster per chromosome with hierarchical clustering. To order the contigs in each cluster, it first finds the maximum spanning tree for the graph corresponding to each cluster. It then finds the longest path in the spanning tree which represents the initial contig ordering. After this, it reinserts contigs which are not part of the initial ordering into the longest path yielding the final contig ordering for each cluster. Once the ordering is computed, it constructs a weighted directed acyclic graph (WDAG) encoding all possible ways in which contigs can be oriented, with the score assigned to each orientation. Finally, it finds the heaviest path through this WDAG describing the optimal orientation assigned to the ordered contigs in each cluster. The primary drawback of LACHESIS is that it needs the number of clusters to be

pre-specified. This method can not be applied to scaffold the contigs of genomes when the number of chromosomes in the organisms are unknown. Kaplan et al. [80] developed a method for scaffolding based on statistical techniques. Their method uses the hierarchical clustering method similar to LACHESIS, but it predicts the number of clusters. The major drawback of their method is that they do not orient the contigs in each cluster, thereby not providing complete information needed for scaffolding. Also, most of the experiments performed using their method used simulated contigs of equal size, except for the scaffolding of chromosome 14. Due to this, it is unclear how their method would perform in the case of long read assemblies where contig lengths can have a large variance. Since both of these methods rely on hierarchical clustering, it is expensive to compute all vs. all link scores for all the contigs, causing scalability issues. The original SALSA1 [85] method first corrects the input assembly, using a lack of Hi-C coverage as evidence of error. It then orients and orders the corrected unitigs to generate scaffolds. Recently, the 3D-DNA [86] method was introduced and demonstrated on a draft assembly of the *Aedes aegypti* genome. 3D-DNA also corrects the errors in the input assembly and then iteratively orients and orders unitigs into a single megascaffold. This megascaffold is then broken into a user-specified number of chromosomes, identifying chromosomal ends based the on Hi-C contact map.

There are several shortcomings common across currently available tools. They require the user to specify the number of chromosomes *a priori*. This can be challenging in novel genomes where no karyotype is available. An incorrect guess often leads to mis-joins that fuse chromosomes. They are also sensitive to input assembly contiguity and Hi-C library variations and require tuning of parameters for each dataset. Inver-

sions are common when the input unitigs are short, as scaffolders determine orientation by maximizing the interaction frequency between unitig ends across all possible orientations [143]. When unitigs are long, few interactions are spanning the full length of the unitig, making the correct orientation apparent from the higher weight of links. However, in the case of short unitigs, interactions are spanning the full length of the unitig, making the correct orientation have a similar weight to incorrect orientations. Biological factors, such as topologically associated domains (TADs) also confound this analysis [144].

SALSA1 [85], addressed some of these challenges, such as not requiring the expected number of chromosomes beforehand and correcting assemblies before scaffolding them with Hi-C data. We showed that SALSA1 worked better than the most widely used method, LACHESIS [143]. However, SALSA1 often did not generate chromosome-sized scaffolds. The contiguity and correctness of the scaffolds depended on the coverage of Hi-C data and required manual data-dependent parameter tuning. Building on this work, SALSA2 does not require manual parameter tuning and can utilize all the contact information from the Hi-C data to generate near optimally sized scaffolds permitted by the data using a novel iterative scaffolding method. In addition to this, SALSA2 enables the use of an assembly graph to guide scaffolding, thereby minimizing errors, particularly orientation errors.

In this work, we introduce SALSA2 - an open source software that combines Hi-C linkage information with the ambiguous-edge information from a genome assembly graph to better resolve unitig orientations. We also propose a novel stopping condition, which does not require an *a priori* estimate of chromosome count, as it naturally stops when the Hi-C information is exhausted. We show that SALSA2 has a fewer number of orienta-

tion, ordering, and chimeric errors across a wide range of assembly contiguities. We also demonstrate robustness to different Hi-C libraries with varying intra-chromosomal contact frequencies. When compared to 3D-DNA, SALSA2 generates more accurate scaffolds across all conditions tested. To our knowledge, this is the first method to leverage assembly graph information for scaffolding Hi-C data.

3.2 Methods

Figure 3.1 (A) shows the overview of the SALSA2 pipeline. SALSA2 begins with a draft assembly is generated from long reads such as Pacific Biosciences [145] or Oxford Nanopore [107]. SALSA2 requires the unitig sequences and, optionally, a GFA-format graph [146] representing the ambiguous reconstructions. Hi-C reads are aligned to the unitig sequences, and unitigs are optionally split in regions lacking Hi-C coverage. A hybrid scaffold graph is constructed using both ambiguous edges from the GFA and edges from the Hi-C reads, scoring edges according to a "best buddy" scheme. Scaffolds are iteratively constructed from this graph using a greedy weighted maximum matching. A mis-join detection step is performed after each iteration to check if any of the joins made during this round are incorrect. Incorrect joins are broken and the edges blacklisted during subsequent iterations. This process continues until the majority of joins made in the prior iteration are incorrect. This provides a natural stopping condition when accurate Hi-C links have been exhausted. Below, we describe each of the steps in detail.

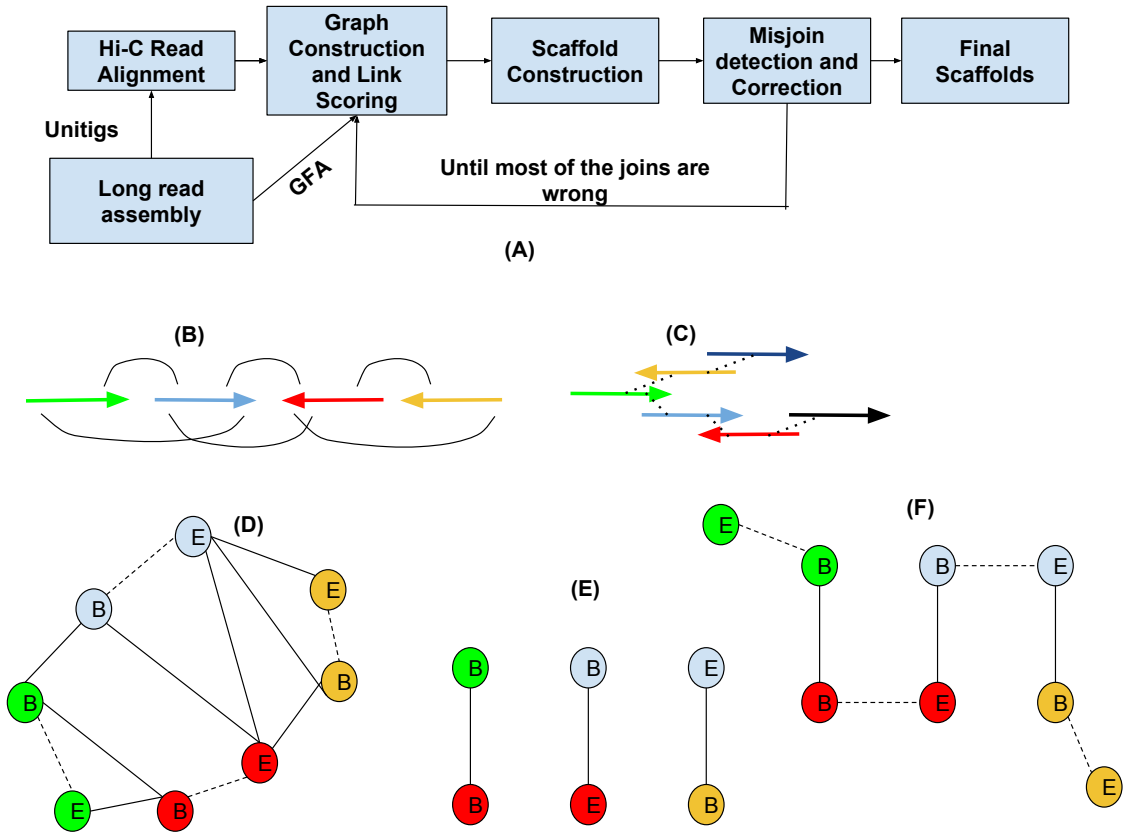


Figure 3.1: (A) Overview of the SALSA2 scaffolding algorithm. (B) Linkage information obtained from the alignment of Hi-C reads to the assembly. (C) The assembly graph obtained from the assembler. (D) A hybrid scaffold graph constructed from the links obtained from the Hi-C read alignments and the overlap graph. Solid edges indicate the linkages between different unitigs and dotted edges indicate the links between the ends of the same unitig. (E) Maximal matching obtained from the graph using a greedy weighted maximum matching algorithm. (F) Edges between the ends of same unitigs are added back to the matching.

3.2.1 Hi-C library preparation

In the Arima-HiC methodology, a sample (cells or tissues) is first crosslinked to preserve the genome conformation. The crosslinked DNA is then digested using restriction enzymes (in this case GATC and GANTC). The single-stranded 5'-overhangs are then filled in causing digested ends to be labeled with a biotinylated nucleotide. Next, spatially proximal digested ends of DNA are ligated, preserving both short- and long-range DNA contiguity. The DNA is then purified and sheared to a size appropriate for Illumina short-read sequencing. After shearing, the biotinylated fragments are enriched to assure that only fragments originating from ligation events are sequenced in paired-end mode via Illumina sequencers to inform DNA contiguity.

3.2.2 Read alignment

Hi-C paired-end reads are aligned to unitigs using the BWA-MEM aligner [78](parameters: -t 12 -B 8) as single-end reads. Reads which align across ligation junctions are chimeric and are trimmed to retain only the start of the read which aligns before the ligation junction. After filtering the chimeric reads, the pairing information is restored. Any PCR duplicates in the paired-end alignments are removed using Picard tools [147]. Read pairs aligned to different unitigs are used to construct the initial scaffold graph. The suggested mapping pipeline is available at http://github.com/ArimaGenomics/mapping_pipeline.

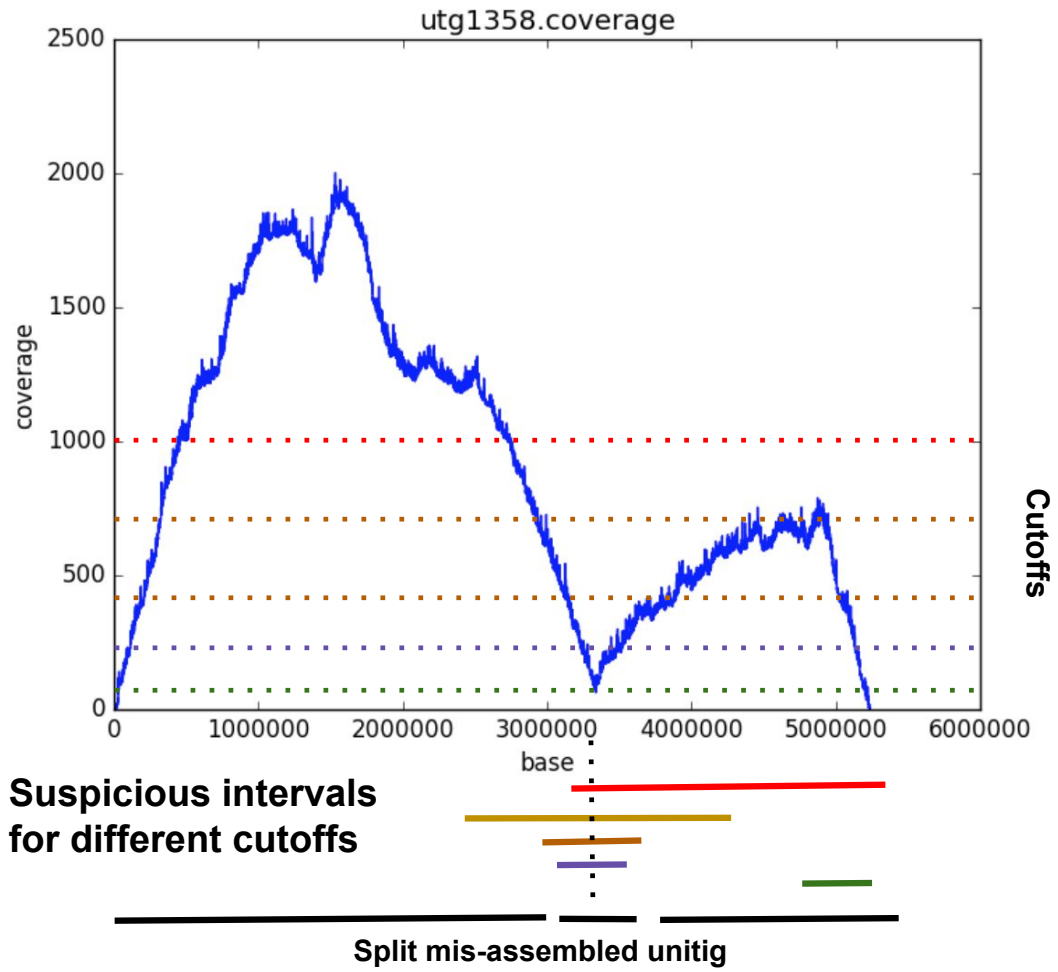


Figure 3.2: Example of the mis-assembly detection algorithm in SALSA2. The plot shows the position on the x-axis and the physical coverage on the y-axis. The dotted horizontal lines show the different thresholds tested to find low physical coverage intervals. The lines at the bottom show the suspicious intervals identified by the algorithm. The dotted line through the intervals shows the maximal clique. The smallest interval (purple) in the clique is identified as mis-assembly, and the unitig is broken in three parts at its boundaries.

3.2.3 Unitig correction

As any assembly is likely to contain mis-assembled sequences, SALSA2 uses the physical coverage of Hi-C pairs to identify suspicious regions and break the sequence at the likely point of mis-assembly. We define the physical coverage of a Hi-C read pair as the region on the unitig spanned by the start of the leftmost fragment and the end of the rightmost fragment. A drop in physical coverage indicates a likely assembly error. We extend the mis-assembly detection algorithm from SALSA which split a unitig when a fixed minimum coverage threshold was not met. A drawback of this approach is that coverage can vary, both due to sequencing depth and variation in Hi-C link density.

Figure 3.2 sketches the new unitig correction algorithm implemented in SALSA2. Instead of a single coverage threshold used in SALSA1, a set of suspicious intervals is found with a sweep of thresholds. Using the collection of intervals as an interval graph, we find the maximal clique. This can be done in $O(NlogN)$ time, where N is the number of intervals. For any clique of a minimum size, the region between the start and end of the smallest interval in the clique is flagged as a mis-assembly, and the unitig is split into three pieces — the sequence to the left of the region, the junction region itself, and the sequence to the right of the region.

3.2.4 Assembly graph construction

For our experiments, we use the unitig assembly graph produced by Canu [148] (Figure 3.1(C)), as this is the more conservative graph output. SALSA2 requires only a GFA format [146] representation of the assembly. Since most long read genome as-

semblers such as FALCON [149], miniasm [146], Canu [148], and Flye [150] provide assembly graphs in GFA format, their output is compatible with SALSA2 for scaffolding.

3.2.5 Scaffold graph construction

The scaffold graph is defined as $G(V, E)$, where nodes V are the ends of unitigs and edges E are derived from the Hi-C read mapping (Figure 3.1B). The idea of using unitig ends as nodes is similar to that used by the string graph formulation [135].

Modeling each unitig as two nodes allow a pair of unitigs to have multiple edges in any of the four possible orientations (forward-forward, forward-reverse, reverse-forward, and reverse-reverse). The graph then contains two edge types - one explicitly connects two different unitigs based on Hi-C data, while the other implicitly connects the two ends of the same unitig.

As in SALSA1, we normalize the Hi-C read counts by the frequency of restriction enzyme cut sites in each unitig. This normalization reduces the bias in the number of shared read pairs due to the unitig length as the number of Hi-C reads sequenced from a particular region are proportional to the number of restriction enzyme cut sites in that region. For each unitig, we denote the number of times a cut site appears as $C(V)$. We define edges weights of G as:

$$W(u, v) = \frac{N(u, v)}{C(u) + C(v)}$$

where $N(u, v)$ is the number of Hi-C read pairs mapped to the ends of the unitigs u and v.

We observed that the globally highest edge weight does not always capture the correct orientation and ordering information due to variations in Hi-C interaction frequencies within a genome. To address this, we defined a modified edge ratio, similar to the one described in [86], which captures the relative weights of all the adjacent edges for a particular node.

The best buddy weight $BB(u, v)$ is the weight $W(u, v)$ divided by the maximal weight of any edge incident upon nodes u or v, excluding the (u, v) edge itself. Computing best buddy weight naively would take $O(|E|^2)$ time and is computationally prohibitive since the graph, G, is usually dense. If the maximum weighted edge incident on each node is stored with the node, the running time for the computation becomes $O(|E|)$. We retain only edges where $BB(u, v) > 1$. This keeps only the edges which are the best incident edge on both u and v. Once used, the edges are removed from subsequent iterations. Thus, the most confident edges are used first, but initially, low scoring edges can become best in subsequent iterations.

For the assembly graph, we define a similar ratio. Since the edge weights are optional in the GFA specification and do not directly relate to the proximity of two unitigs on the chromosome, we use the graph topology to establish this relationship. Let $\bar{u}$ denote the reverse complement of the unitig u. Let $\sigma(u, v)$ denote the length of shortest path between u and v. For each edge (u, v) in the scaffold graph, we find the shortest path between unitigs u and v in every possible orientation, that is, $\sigma(u, v)$, $\sigma(u, \bar{v})$, $\sigma(\bar{u}, v)$ and $\sigma(\bar{u}, \bar{v})$. With this, the score for a pair of unitigs is defined as follows:

$$Score(u,v) = \frac{\min\limits_{x' \in \{u,\bar{u}\}-\{x\}, y' \in \{v,\bar{v}\}-\{y\}} \sigma(x',y')}{\min\limits_{x \in \{u,\bar{u}\}, y \in \{v,\bar{v}\}} \sigma(x,y)}$$

where x and y are the orientations in which u and v are connected by the shortest path in the assembly graph. Essentially, $Score(u,v)$ is the ratio of the length of the second shortest path to the length of the shortest path in all possible orientations. Once again, we retain edges where $Score(u,v) > 1$. If the orientation implied by the assembly graph differs from the orientation implied by the Hi-C data, we remove the Hi-C edge and retain the assembly graph edge (Figure 3.1D). Computing the score graph requires $|E|$ shortest path queries, yielding total runtime of $O(|E| * (|V| + |E|))$ since we do not use the edge weights.

3.2.6 Unitig layout

Once we have the hybrid graph, we lay out the unitigs to generate scaffolds. Since there are implicit edges in the graph G between the beginning and end of each unitig, the problem of computing a scaffold layout can be modeled as finding a weighted maximum matching in a general graph, with edge weights being our ratio weights. If we find the weighted maximum matching of the non-implicit edges (that is, edges between different unitigs) in the graph, adding the implicit edges to this matching would yield a complete traversal. However, adding implicit edges to the matching can introduce a cycle. Such cycles are removed by removing the lowest weight non-implicit edge. Computing a maximal matching takes $O(|E||V|^2)$ time [151]. We iteratively find a maximum matching in the graph by removing nodes found in the previous iteration. Using the optimal maximum

matching algorithm, this would take $O(|E||V|^3)$ time, which would be extremely slow for large graphs. Instead, we use a greedy maximal matching algorithm which is guaranteed to find a matching within 1/2-approximation of the optimum [152]. The greedy matching algorithm takes $O(|E|)$ time, thereby making the total runtime $O(|V||E|)$. The algorithm for unitig layout is sketched in Algorithm 1. Figure 3.1(D - F) show the layout on an example graph. Contigs which were not scaffolded are inserted in the large scaffolds with the method used in SALSA1.

Algorithm 1 Unitig Layout Algorithm

```
E : Edges sorted by the best buddy weight
M : Set to store maximal matchings
G : The scaffold graph
while all nodes in G are not matched do
  M* = {}
  for e ∈ E sorted by best buddy weights do
    if e can be added to M* then
      M* = M* ∪ e
    end if
  end for
  M = M ∪ M*
  Remove nodes and edges which are part of M* from G
end while
```

3.2.7 Iterative mis-join correction

Since the unitig layout is greedy, it can introduce errors by selecting a false Hi-C link which was not eliminated by our ratio scoring. These errors propagate downstream, causing large chimeric scaffolds and chromosomal fusions. We examine each join made within all the scaffolds in the last iteration for correctness. Any join with low spanning Hi-C support relative to the rest of the scaffold is broken, and the links are blacklisted for further iterations.

We compute the physical coverage spanned by all read pairs aligned in a window of size w around each join. For each window, w, we create an auxiliary array, which stores -1 at position i if the physical coverage is higher than some cutoff δ and 1, otherwise. We then find the maximum sum subarray in this auxiliary array, since it captures the longest stretch of low physical coverage. If the position being tested for mis-join lies within the region spanned by the maximal clique generated with the maximum sum subarray intervals for different cutoffs (Figure 3.2), the join is marked as incorrect. The physical coverage can be computed in $O(w + N)$ time, where N is the number of read pairs aligned in window w. The maximum sum subarray computation takes $O(w)$ time. If K is the number of cutoffs(δ) tested for the suspicious join finding, then the total runtime of mis-assembly detection becomes $O(K(N + 2 * w))$. The parameter K controls the specificity of the mis-assembly detection, thereby avoiding false positives. The algorithm for mis-join detection is sketched in Algorithm 2. When the majority of joins made in a particular iteration are flagged as incorrect by the algorithm, SASLA2 stops scaffolding and reports the scaffolds generated in the penultimate iteration as the final result.

3.3 Results

3.3.1 Dataset description

We created artificial assemblies, each containing unitigs of the same size, by splitting the GRCh38 [153] reference into fixed sized unitigs of 200 to 900 kbp. This gave us eight assemblies. The assembly graph for each input is built by adding edges for any adjacent unitigs in the genome.

Algorithm 2 Misjoin detection and correction algorithm

Cov : Physical coverage array for a window size w around a scaffold join at position p on a scaffold
A : Auxiliary array
I : Maximum sum subarray intervals
for $\delta \in$ {min_coverage, max_coverage} **do**
  if $Cov[i] \leq \delta$ **then**
    $A[i] = 1$
  else
    $A[i] = -1$
  end if
  $s_\delta, e_\delta =$ maximum_sum_subarray(A)
  $I = I \cup \{s_\delta, e_\delta\}$
end for
$s, e =$maximal_clique_interval(I)
if $p \in \{s, e\}$ **then**
  Break the scaffold at position p
end if

For real data, we use the recently published NA12878 human dataset sequenced with Oxford Nanopore [154] and assembled with Canu [148]. We use a Hi-C library from Arima Genomics (Arima Genomics, San Diego, CA) sequenced to 40x coverage (SRX3651893). We compare results with the original SALSA (https://github.com/machinegun/SALSA/tree/833fb11), SALSA2 with and without the assembly graph input (https://github.com/machinegun/SALSA/tree/eb9aeec), and 3D-DNA (https://github.com/theaidenlab/3d-dna/tree/745779b). We did not compare our results with LACHESIS because it is no longer supported and is outperformed by 3D-DNA [86]. SALSA2 was run using default parameters, except for graph incorporation, as listed. For 3D-DNA, alignments were generated using the Juicer alignment pipeline [74] with defaults (-m haploid -t 15000 -s 2), except for mis-assembly detection, as listed. The chromosome number was set to 23 for all experiments. The genome size of 3.2 Gbp was used

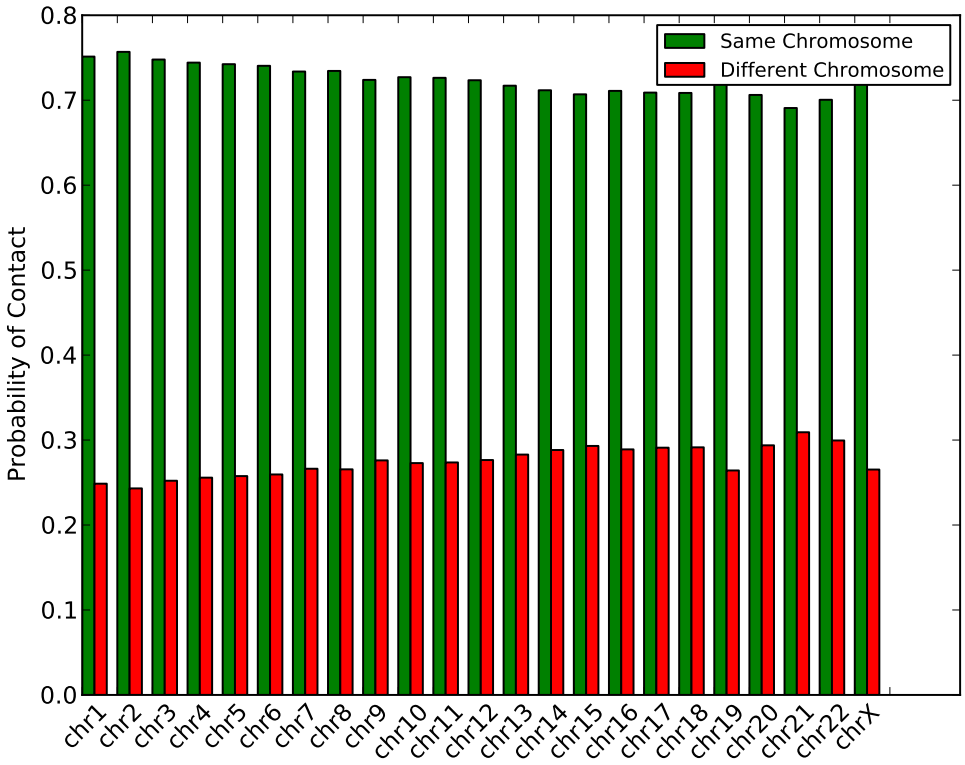


Figure 3.3: The probability of contact calculated based on read mapping to GRCh38 reference.

for contiguity statistics for all assemblies.

For evaluation, we also used the GRCh38 reference to define a set of true and false links from the Hi-C graph. We mapped the assembly to the reference with MUMmer3.23 (nucmer -c 500 -l 20) [155] and generated a tiling using MUMmer's show-tiling utility. For this "true link" dataset, any link joining unitigs in the same chromosome in the correct orientation was marked as true. This also gives the true unitig position, orientation, and chromosome assignment. We masked sequences in GRCh38 which matched known structural variants from a previous assembly of NA12878 [120] to avoid counting true variations as scaffolding errors.

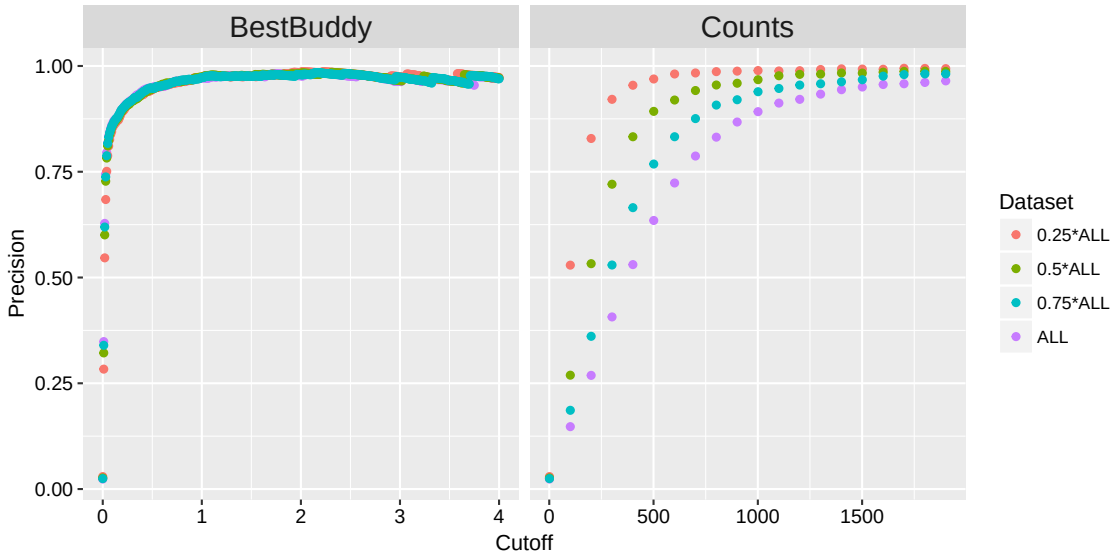


Figure 3.4: Precision at different cutoffs for Hi-C links. The plot on the left shows the curve for the SALSA2 best buddy weight cutoffs and the plot on the right shows the curve for a fixed Hi-C pair count cutoff, used in SALSA1, across changing coverage.

3.3.2 Contact probability of Hi-C data

We aligned Hi-C reads from NA12878, a human genome used in the 1000 Genomes project [156], to the GRCh38 human reference genome [153] using BWA-MEM (version - 0.7.13) [78] with default parameters. If both mates in the read pair align to the same chromosome, it implies an intrachromosomal contact. For each chromosome, we count how many read pairs have both mates mapped to that chromosome and how many reads have just one of the mates mapped to that chromosome. Using this information, we compute the intrachromosomal and interchromosomal contact probability for each chromosome. It can be seen from Figure 3.3 that the probability of intrachromosomal contact is much higher than that of interchromosomal contact.

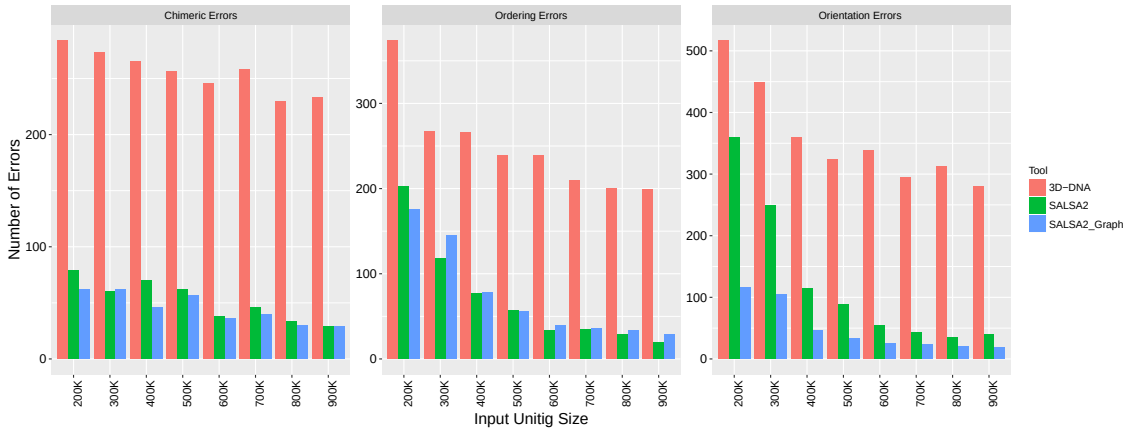


Figure 3.5: Comparison of orientation, ordering, and chimeric errors in the scaffolds produced by SALSA2 and 3D-DNA on the simulated data. As expected, the number of errors for all error types decrease with increasing input unitig size. Incorporating the assembly graph reduces error across all categories and most assembly sizes, with the largest decrease seen in orientation errors. SALSA2 utilizing the graph has 2-4 fold fewer errors than 3D-DNA.

3.3.3 Scoring effectiveness

For correct scaffolding, we want to filter false edges and retain only the correct linkage information between pairs of unitigs. Our previous algorithm used a fixed, user-defined minimum for edges connecting a pair of unitigs. The drawback of a fixed cutoff is that it cannot handle variations in coverage within the assembly and varies between any pair of sequencing datasets. To compare the scoring methods, we down-sample the alignments into three different sets with 0.25, 0.5 and 0.75 of the original coverage and computed the precision of filtering based on the ratio score and a fixed threshold. The precision remained almost constant for the ratio cutoff on all datasets, whereas the precision changes rapidly for different coverages and a fixed threshold (Figure 3.4).

3.3.4 Evaluation on simulated unitigs

3.3.4.1 Assembly correction

We simulated assembly error by randomly joining 200 pairs of unitigs from each simulated assembly. All erroneous joins were made between unitigs that are more than 10 Mbp apart or were assigned to different chromosomes in the reference. The remaining unitigs were unaltered. We then aligned the Arima-HiC data and ran our assembly correction algorithm. When the algorithm marked a mis-join within 20 kbp of a true error we called it a true positive; otherwise, we called it a false positive. Any unmarked error was called a false negative. The average sensitivity overall simulated assemblies was 77.62% and the specificity was 86.13%. The sensitivity was highest for larger unitigs (50% for 200 kbp versus >90% for untigs greater than 500 kbp) implying that our algorithm can accurately identify errors in large unitigs, which can have a negative impact on the final scaffolds if not corrected. Although we used a cutoff of 20kbp to evaluate sensitivity and specificity, most of the predicted locations of misassembly were within 5kbp from the true misassembly location (Figure 3.6).

3.3.4.2 Scaffold mis-join validation

As before, we simulated erroneous scaffolds by joining unitigs which were not within 10 Mbp in the reference or were assigned to different chromosomes. Rather than pairs of unitigs, each erroneous scaffold joined ten unitigs, and we generated 200 such erroneous scaffolds. The remaining unitigs were correctly scaffolded (ten unitigs per

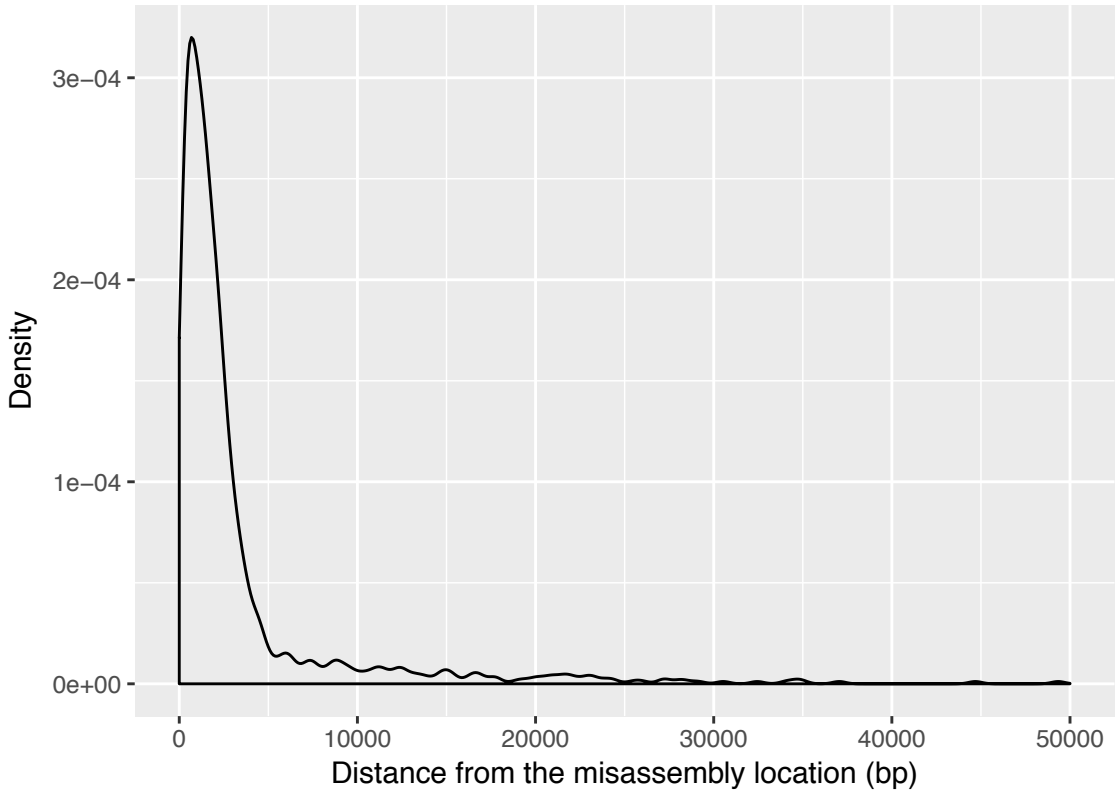


Figure 3.6: Distribution of the distance between the predicted misassembly location to the actual misassembly location in the simulated data.

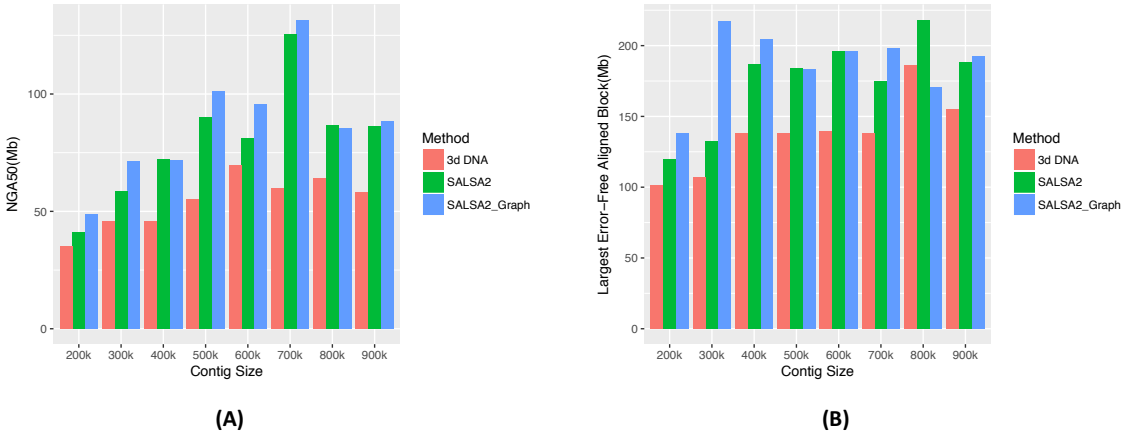


Figure 3.7: (A) NGA50 statistic for different input unitig sizes and (B) The length of longest error-free block for different input unitig sizes. Once again, the assembly graph typically increases both the NGA50 and the largest correct block.

scaffold) based on their location in the reference. The average sensitivity was 68.89% and specificity was 100% (no correct scaffolds were broken). Most of the un-flagged joins occurred near the ends of scaffolds and could be captured by decreasing the window size. Similar to assembly correction, we observed that sensitivity was highest with larger input unitigs. This evaluation highlights the accuracy of the mis-join detection algorithm to avoid over-scaffolding and provide a suitable stopping condition.

3.3.4.3 Scaffold accuracy

We evaluated scaffolds across three categories of error: orientation, order, and chimera. An orientation error occurs whenever the orientation of a unitig in a scaffold differs from that of the scaffold in the reference. An ordering error occurs when a set of three unitigs adjacent in a scaffold have non-monotonic coordinates in the reference. A chimera error occurs when any pair of unitigs adjacent in a scaffold align to different chromosomes in the reference. We broke the assembly at these errors and computed corrected scaffold lengths and NGA50 (analogous to the NGA50 defined by Salzberg et al. [157]). This statistic corrects for large but incorrect scaffolds which have a high NG50 but are not useful for downstream analysis because of errors. We did not include SALSA1 in the comparison because for small contig sizes (200 kbp to 500 kbp), none of the scaffolds contained more than two contigs. For larger sizes (600 kbp to 900 kbp), the contiguity widely varied depending upon the minimum confidence parameter for accepting links between contigs.

Hi-C scaffolding errors, particularly orientation errors, increased with decreasing

assembly contiguity. We evaluated scaffolding methods across a variety of simulated unitig sizes. Figure 3.5 shows the comparison of these errors for 3D-DNA, SALSA2 without the assembly graph, and SALSA2 with the graph. SALSA2 produced fewer errors than 3D-DNA across all error types and input sizes. The number of correctly oriented unitigs increased significantly when assembly graph information was integrated with the scaffolding, particularly for lower input unitig sizes (Figure 3.5). For example, at 400 kbp, the orientation errors with the graph were comparable to the orientation errors of the graph-less approach at 900 kbp. The NGA50 for SALSA2 also increased when assembly graph information was included (Figure 3.7). This highlights the power of the assembly graph to improve scaffolding and correct errors, especially on lower contiguity assemblies. This also indicates that generating a conservative assembly, rather than maximizing contiguity, can be preferable for input to Hi-C scaffolding.

3.3.5 Evaluation on NA12878

Table 3.1 lists the metrics for NA12878 scaffolds. We include an idealized scenario, using only reference-filtered Hi-C edges for comparison. As expected, the scaffolds generated using only true links had the highest NGA50 value and longest error-free scaffold block. SALSA2 scaffolds were more accurate and contiguous than the scaffolds generated by SALSA1 and 3D-DNA, even without the use of the assembly graph. The addition of the graph further improved the NGA50 and longest error-free scaffold length.

We also evaluated the assemblies using Feature Response Curves (FRC) based on scaffolding errors [158]. An assembly can have a high raw error count but still be of high

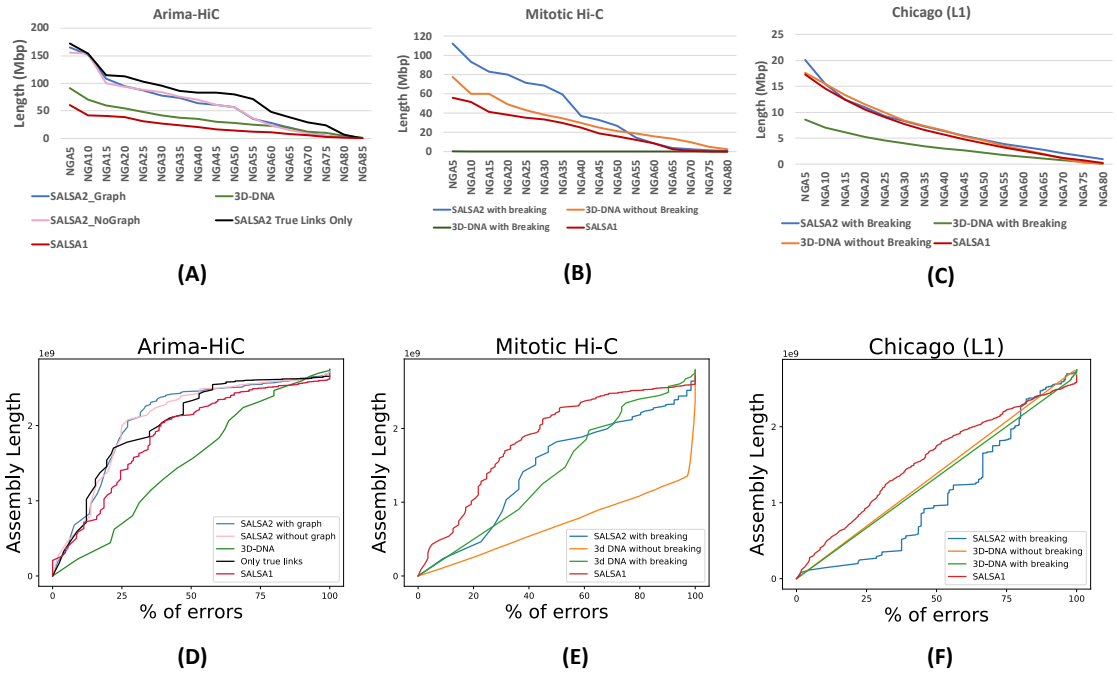


Figure 3.8: Contiguity plot for scaffolds generated with (A) standard Arima-HiC data (B) mitotic Hi-C data and (C) Chicago data. The X-axis denotes the NGAX statistic, and the Y-axis denotes the corrected block length to reach the NGAX value. SALSA2 results were generated using the assembly graph unless otherwise noted. Feature Response Curve for (D) assemblies obtained from unitigs as input (E) assemblies obtained from mitotic Hi-C data and (F) assemblies obtained using Dovetail Chicago data. The best assemblies lie near the top left of the plot, with the largest area under the curve. The FRC for 3D-DNA scaffolds with Chicago input is a straight line because 3D-DNA generated a single 2.7 Gbp super-scaffold which contained the majority of the genome sequence.

Dataset	Method	NG50(Mbp)	NGA50(Mbp)	Longest Chunk (Mbp)	Orientation Errors	Ordering Errors	Chimeric Errors
Arima-HiC	SALSA2 true links	83.31	79.48	172.19	78	101	0
	SALSA2 w graph	125.34	57.20	165.11	156	289	142
	SALSA2 wo graph	101.96	56.84	155.68	168	302	152
	3D-DNA	137.88	28.61	130.88	233	405	178
	SALSA1	19.09	14.81	73.14	99	176	96
Mitotic Hi-C	SALSA2 w graph	69.23	26.46	145.53	117	98	58
	SALSA1	27.88	15.62	85.71	142	78	120
	3D-DNA w correction	16.34	0.064	0.96	12017	11687	7217
	3D-DNA wo correction	141.18	21.47	84.00	345	320	163
Chicago	SALSA2 w graph	6.15	4.63	34.60	56	72	128
	SALSA1	5.21	3.94	34.60	83	21	187
	3D-DNA w correction	2641.31	2.62	12.76	244	186	1550
	3D-DNA wo correction	1684.92	4.52	34.60	119	100	711
Illumina Assembly	SALSA2 w graph	90.26	9.04	31.84	1484	2348	615
	SALSA2 wo graph	89.12	4.06	26.22	2196	2318	723
	3D-DNA w correction	134.65	8.12	48.33	1545	2126	529
	3D-DNA wo correction	137.86	9.53	42.63	1363	2002	308

Table 3.1: Assembly scaffold and correctness statistics for NA12878 assemblies scaffolded with different Hi-C libraries. The NG50 of human reference GRCh38 is 145 Mbp. The ratio between NG50 and NGA50 represents how many erroneous joins affect large scaffolds in the assembly. A high ratio between NGA50 and NG50 indicates a more accurate assembly. We observe that 3D-DNA mis-assembly detections shears the input with both the mitotic Hi-C and Chicago data so we include results both with and without this assembly correction. In case of Chicago data, 3D-DNA generates a large super-scaffold containing more than 50% of the genome, giving a very high NG50 but a poor NGA50 and ratio.

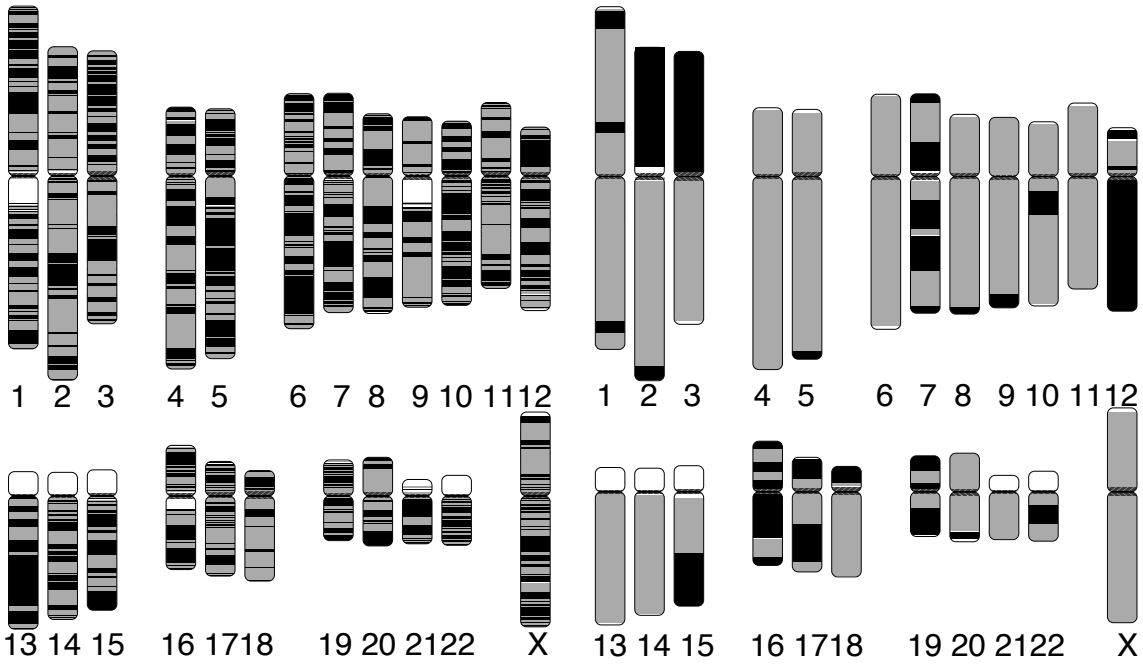


Figure 3.9: Chromosome ideogram generated using the coloredChromosomes package. Each color switch denotes a change in the aligned sequence, either due to large structural error or the end of a unitig/scaffold. Left: input unitigs aligned to the GRCh38 reference genome. Right: SALSA2 scaffolds aligned to the GRCh38 reference genome. More than ten chromosomes are in a single scaffold. Chromosomes 1 and 7 are more fragmented due to scaffolding errors which break the alignment.

quality if the errors are restricted to only short scaffolds. FRC captures this by showing how quickly error is accumulated, starting from the largest scaffolds. Figure 3.8(D) shows the FRC for different assemblies, where the X-axis denotes the cumulative % of assembly errors, and the Y-axis denotes the cumulative assembly size. The assemblies with more area under the curve accumulate fewer errors in larger scaffolds and hence are more accurate. SALSA2 scaffolds with and without the graph have similar areas under the curve and closely match the curve of the assembly using only true links. The 3D-DNA scaffolds have the lowest area under the curve, implying that most errors in the assembly occur in the long scaffolds. The lower NGA50 value confirms this for the 3D-DNA assembly (Table 3.1).

Apart from the correctness, SALSA2 scaffolds were highly contiguous and reached an NG50 of 125 Mbp (cf. GRCh38 NG50 of 145 Mbp). Figure 3.9 shows the alignment ideogram for the input unitigs as well as the SALSA2 assembly. Every color change indicates an alignment break, either due to error or due to the end of a sequence. The input unitigs are fragmented with multiple unitigs aligning to the same chromosome, while the SALSA2 scaffolds are highly contiguous and span entire chromosomes in many cases. Figure 3.8(A) shows the contiguity plot with corrected NG stats. As expected, the assembly generated with only true links has the highest values for all NGA stats. The curve for SALSA2 assemblies with and without the assembly graph closely matches this curve, implying that the scaffolds generated with SALSA2 are approaching the optimal assembly of this Arima-HiC data.

We also evaluated the ability of scaffolding short read assemblies for both 3D-DNA and SALSA2. We did not include SALSA1 in this comparison because it is not

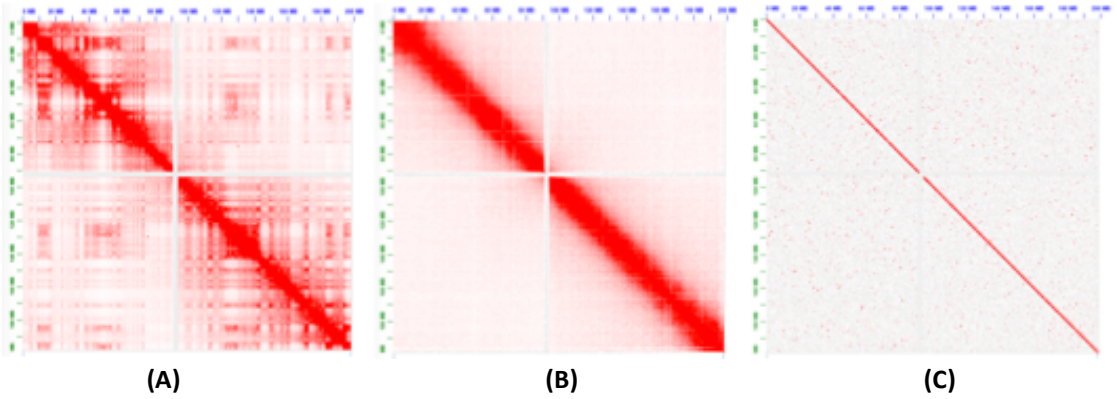


Figure 3.10: Contact map of Hi-C interactions on Chromosome 3 generated by the Juicebox software [75]. The cells sequenced in (A) normal conditions, (B) during mitosis, and (C) Dovetail Chicago

designed to scaffold short read assemblies. We observed that adding assembly graph to scaffolding significantly reduced the number of orientation errors for SALSA2, increasing the scaffold NG50 almost two-fold. When compared to 3D-DNA with or without input assembly correction, SALSA2 with the assembly graph generates scaffolds of comparable sizes with a similar number of errors as 3D-DNA.

To understand the computational requirements of SALSA2 and 3D-DNA, we computed the CPU runtime and memory usage for both the methods while scaffolding long and short read assemblies with Arima-HiC data. While scaffolding the long-read assembly SALSA2 required 11.44 CPU hours and 21.43 Gb peak memory, 3D-DNA required 318 CPU hours and 64.66 Gb peak memory. While scaffolding the short-read assembly, SALSA2 required 36.8 CPU hours and 61.8 Gb peak memory compared to 2080 CPU hours and 48.04 Gb peak memory needed by 3D-DNA.

3.3.6 Robustness to input library

We next tested scaffolding using two libraries with different Hi-C contact patterns. The first, from [159], is sequenced during mitosis. This removes the topological domains and generates fewer off-diagonal interactions. The other library was from [71], are *in vitro* chromatin sequencing library (Chicago) generated by Dovetail Genomics (L1). It also removes off-diagonal matches but has shorter-range interactions, limited by the size of the input molecules. As seen from the contact map in Supplementary Figure S2, both the mitotic Hi-C and Chicago libraries follow different interaction distributions than the standard Hi-C (Arima-HiC in this case). We ran SALSA2 with defaults, and 3D-DNA with both the assembly correction turned on and off.

For mitotic Hi-C data, we observed that the 3D-DNA mis-assembly correction algorithm sheared the input assembly into small pieces, which resulted in more than 12,000 errors and more than half of the unitigs incorrectly oriented or ordered. Without mis-assembly correction, the 3D-DNA assembly has a higher number of orientation (345 vs. 117) and ordering (320 vs. 98) errors compared to SALSA2. The feature response curve for the 3D-DNA assembly with breaking is almost a diagonal (Figure 3.8(E)) because the sheared unitigs appeared to be randomly joined. SALSA2 scaffolds contain long stretches of correct scaffolds compared to 3D-DNA with and without mis-assembly correction (Figure 3.8(B)). SALSA1 scaffolds had a similar error count to SALSA2 but were less contiguous.

For the Chicago libraries, 3D-DNA mis-assembly detection once again sheared the input unitigs. It generated a single 2.7 Gbp scaffold and was unable to split it into the

requested number of chromosomes. 3D-DNA uses signatures of chromosome ends [86] to identify break positions which are not present in Chicago data. As a result, it generated more chimeric joins compared to SALSA2 (1,550 vs. 128 errors). However, the number of order and orientation errors was similar across the methods. Even in the single large scaffold generated by 3D-DNA, the sizes of the correctly oriented and ordered blocks were smaller than SALSA2 (Figure 3.8(C)). Since Chicago libraries do not provide chromosome-spanning contact information for scaffolding, the NG50 value for SALSA2 is 6.15 Mbp, comparable to the equivalent coverage assembly (50% L1+L2) in [71] but much smaller than Hi-C libraries. Interestingly, SALSA1 was able to generate scaffolds of similar contiguity to SALSA2, which can be attributed to the lack of long-range contact information in the library. SALSA2 is robust to changing contact distributions. In the case of Chicago data, it produced a less contiguous assembly due to the shorter interaction distance. However, it avoids introducing false joins, unlike 3D-DNA, which appears tuned for a specific contact model.

3.4 Conclusion

In this work, we present the first Hi-C scaffolding method that integrates an assembly graph to produce high-accuracy, chromosome-scale assemblies. Since long read assemblies were used, most of the issues cause by repeats were solved. Our experiments on both simulated and real sequencing data for the human genome demonstrate the benefits of using an assembly graph to guide scaffolding. We also show that SALSA2 outperforms alternative Hi-C scaffolding tools on assemblies of varied contiguity, using multiple Hi-C

library preparations.

Hi-C scaffolding has been historically prone to inversion errors when the input assembly is highly fragmented. The integration of the assembly graph with the scaffolding process can overcome this limitation. Orientation errors introduced in the assembly and scaffolding process can lead to incorrect identification of structural variations. On simulated data, more than 50% of errors were due to inversions, and integrating the assembly graph reduced these by as much as 3 to 4 fold. We did not observe as much improvement with the NA12878 test dataset because the contig NG50 was much higher than in the simulation. However, it is not always possible to assemble multi-megabase contigs. In such cases, the assembly graph is useful for limiting Hi-C errors.

Existing Hi-C scaffolding methods also require an estimate for the number of chromosomes for a genome. This is implicitly taken to be the desired number of scaffolds to output. However, as demonstrated by the Chicago, mitotic, and replicate [160] Hi-C libraries, the library, as well as the genome, influences the maximum correct scaffold size. It can be impractical to sweep over hundreds of chromosome values to select a "best" assembly. Since SALSA2's mis-join correction algorithm stops scaffolding after the useful linking information in a dataset is exhausted, no chromosome count is needed as input. As the Genome10K consortium [161] and independent scientists begin to sequence novel lineages in the tree of life, it may be impractical to generate physical or genetics maps for every organism. Thus, Hi-C sequencing combined with SALSA2 presents an economical alternative for the reconstruction of chromosome-scale assemblies.

Chapter 4: Tuning Hi-C library preparation for accurate scaffolding

4.1 Introduction

In the previous chapter, we demonstrated the utility of using Hi-C data to obtain chromosome scale assemblies of large genomes. Hi-C technology was initially developed to interrogate the 3D architecture of the genome inside the cell nucleus. After the invention of Hi-C, researchers realized its potential as a tool for scaffolding because of the long-range contacts present in the Hi-C library. However, along with the signal corresponding to the proximity of genomic loci, Hi-C library also contains some confounding biological signal resulting from telomeric clustering or chromatin loopings. Thus, not all the contacts implied by the Hi-C data are useful for chromosome scale scaffolding and may cause scaffolding errors. Figuring out which of the long-range contacts are useful for scaffolding is challenging from a computational point of view. Thus, developing a library that would strictly imply the proximity of genomic loci is highly desired for accurate chromosome-scale assembly. Such a library will be specially tuned for the scaffolding purpose. Another reason for exploring the possibility of tuning the Hi-C library for scaffolding is the cost of sequencing. In large-scale genome sequencing projects, to obtain 'gold standard' genome assemblies, usually, data from different sequencing technologies are used, incurring significant sequencing costs. If a single sequencing library

can help at multiple steps of genome assembly, it would bring down sequencing cost. In this project, we explored the feasibility of a one-pot library which would have uniform sequencing coverage: a property essential for base level polishing of an assembly, along with yielding accurate chromosome scale scaffolds. We specifically evaluated Hi-C libraries for different metrics, namely, scaffolding accuracy, base level polishing accuracy and haplotype phasing accuracy on the Human and Chicken genomes and showed that such a modified Hi-C library could provide benefit at different stages of genome finishing process.

4.2 Methods

4.2.1 Hi-C libraries and read Mapping

We obtained two types of Hi-C libraries for Human and Chicken genome samples from Arima Genomics, San Diego. One library referred to as Hi-C henceforth was prepared using the standard Hi-C library preparation [162]. The other library referred to as Prime henceforth was prepared using a proprietary library preparation by Arima Genomics that aims to reduce the fraction of interchromosomal interactions and increase coverage uniformity. Different Hi-C scaffolding methods require Hi-C read alignments to the assembly in different formats. To generate a Hi-C contact map as input to 3D-DNA, we used Juicer's alignment pipeline, by providing appropriate restriction enzyme as an input. To generate alignments for SALSA2, we used Arima's Mapping pipeline. This pipeline first maps reads to the assembly using BWA-MEM and then filters out alignments for Hi-C experimental artifacts such as fragment size, digestion and ligation effi-

ciency, mappability, sequencing coverage, GC content, and restriction site density. The final output is then converted to bed format.

4.2.2 Scaffolding with HiC data

Once the alignments were generated, the assemblies were scaffolded using 3D-DNA and SALSA2. To run 3D-DNA, we turned the input assembly correction option off as it is tuned to a specific Hi-C contact distribution. We ran SALSA2 with default parameters. We then converted the scaffolds obtained by both the methods into an AGP format to interpret the orientation and ordering of contigs along scaffolds. We then used this AGP file to compute scaffolding errors. We quantified the accuracy of scaffolds based on orientation, ordering, and chimeric errors by comparing the orientation and ordering of contigs generated by each scaffolding method to the orientation and ordering of contigs implied by aligning them to the reference.

4.2.3 Haplotype phasing

We first aligned reads from each library (Hi-C, Prime, and Illumina) to GRCh38 reference genome using BWA-MEM. With these alignments, variants were called using FreeBayes [163] in diploid mode. These variants were then phased using the HapCUT2 [130] phasing algorithm. We used different combinations of variants and linking information for phasing. We used 1) Illumina variants with Hi-C alignments, 2) Illumina variants with Prime alignments, 3) Hi-C variants with Hi-C alignments and 4) Prime variants with Prime alignments. Once the phasing was obtained, phased haplotypes were

compared with true haplotypes to evaluate the phasing accuracy.

4.3 Results

4.3.1 Library characteristics

Organisms	Library	Intrachromosomal	Interchromosomal	Short Range	Long Range	Sequence Coverage
Human	HiC	84.25%	15.74%	39%	61%	96.34%
	Prime	88.08%	11.91%	60.43%	39.56%	96.45%
Chicken	HiC	82.51%	17.84%	42.89%	57.10%	95.26%
	Prime	90.03%	10.07%	71.98%	28.01%	97.77%

Table 4.1: Alighment statistics of Hi-C and Prime libraries when aligned to Human and Chicken reference genomes.

To understand the difference between the type of contact information provided by both the libraries, we aligned them to the respective reference genomes and analyzed the alignment patterns. We observed that the Prime libraries provide more intra-chromosomal contacts compared to Hi-C library (88.08% vs. 84.25% for human and 90.03% vs. 82.51% for chicken)(Table 4.1). This property is useful for scaffolding since inter-chromosomal contacts confound the scaffolding algorithm into joining contigs belonging to different chromosomes. To further understand the extent of intra-chromosomal interactions, we calculated the distance between the reads in each pair aligned to the same chromosome. We call an interaction short if the distance is less than 20 kbp and long, otherwise. We observed that for both human and chicken libraries, Prime had more short-range interactions compared to Hi-C. This seems counterintuitive since we would desire more long-range interactions for long-range scaffolding. We show in the next section that fewer long-range interactions do not hurt scaffold contiguity and we obtain similar

contiguity with Prime and Hi-C libraries.

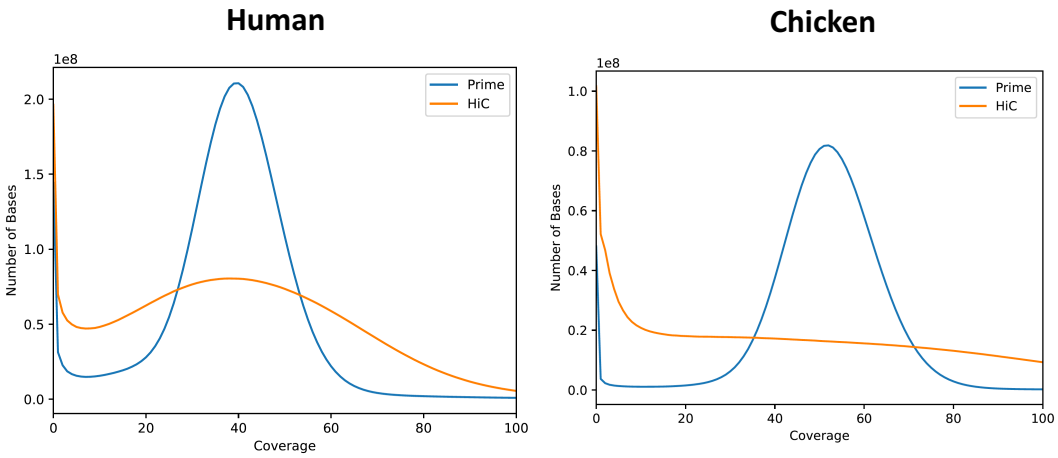


Figure 4.1: Density plots for Hi-C and Prime library per-base coverage for human and chicken genomes when aligned to the reference genome.

One more important library property of the libraries we quantified is the number of bases covered with the Hi-C and Prime library reads when aligned to the reference genomes. This is particularly useful because higher and more uniform the per-base coverage of a library is, better base-level polishing accuracy is obtained with that library. When we counted just the percentage of bases with non-zero sequencing coverage, it was very similar for both Hi-C and Prime libraries. However, when we checked the distribution of coverage values at each base in the reference, the distributions look very different for both the libraries (Figure 4.1). For both the genomes, Prime library coverage follows a normal distribution; implying that all bases in the reference genomes have similar sequencing coverage which is beneficial for base level polishing. On the contrary, the Hi-C library has an exponential distribution implying that bases in the reference genomes have extremely variable coverage. This means that although both the libraries have similar physical coverage, the sequence coverage is not uniform in the Hi-C library than the

Prime library. In results, we show that how this difference in the sequencing coverage affects the polishing accuracy.

4.3.2 Scaffolding accuracy with different libraries

We scaffolded contig assemblies of human and chicken genome with both Prime and Hi-C libraries. The human contig assembly was generated with Oxford Nanopore long read data. The chicken contig assembly was generated with PacBio long read data. We assembled both the genomes with Canu assembler. Table 4.2 shows various metrics for scaffolds generated with different libraries. We used two different scaffolding algorithms to generate scaffolds to make sure we are not biasing any library with the method used for scaffolding. For the human genome scaffolds, both SALSA2 and 3D-DNA had similar NG50 for Prime and Hi-C libraries. However, for SALSA2, the NGA50 (analogous to the NGA50 defined by Salzberg et al. [157]) was higher for the Prime library than Hi-C (50.31 Mbp vs. 40.10 Mbp). For 3D-DNA the NGA50 was similar for scaffolds with both the libraries. This implied that the Prime library does not hurt the contiguity for human genome scaffolding although it has much more short-range interactions than Hi-C library. For the chicken genome scaffolding, the NG50 was much higher with Hi-C library for both SALSA2 and 3D-DNA. However, for SALSA2, the NGA50 was comparable with both the libraries. For scaffolding with 3D-DNA, we observed that the NGA50 was significantly lower with Prime libraries. This can be attributed to the fact that 3D-DNA is tuned for a particular Hi-C contact distribution which expects more long-range interactions than what Prime library provides. For all the scaffolds, we observed that the

number of chimeric errors was fewer when scaffolded with Prime libraries because of the fewer inter-chromosomal interactions present in the Prime library. Also in all cases, the length of longest error-free chink was equal or more for scaffolds with Prime library than Hi-C.

Organisms	Method	Library	NG50(Mbp)	NGA50(Mbp)	Largest error free chunk	Orientation Errors	Ordering errors	Chimeric errors
Human	SALSA2	HiC	101.05	40.10	155.87	129	121	137
		Prime	97.92	50.31	169.33	112	119	129
	3D-DNA	HiC	83.93	21.87	128.04	171	133	125
		Prime	90.32	19.15	127.90	184	136	119
Chicken	SALSA2	HiC	68.53	24.19	95.16	60	61	109
		Prime	57.12	23.06	103.89	53	46	58
	3D-DNA	HiC	90.19	22.07	98.73	150	138	295
		Prime	45.17	12.09	110.54	133	125	298

Table 4.2: Assembly scaffold and correctness statistics for Human and Chicken genome assemblies when scaffolded with Hi-C and Prime libraries.

4.3.3 Polishing accuracy

Usually, genomes assembled with long reads have higher rates of short insertions and deletions due to limitations in sequencing accuracy and basecalling algorithmic limitations. To overcome this, high accuracy Illumina data is typically used to 'polish' the assembly which essentially tests the correctness of each base in the assembly using the pileup of the Illumina reads and corrects the base if it does not agree with the consensus implied by the read pileup. Since Prime libraries provide more uniform sequence coverage, the hypothesis is that it would also yield better base level accuracy after polishing. To test this, we aligned both the libraries to the respective scaffolds and used Pilon [164] in the paired-end mode to polish the assemblies based on these alignments. To obtain the base level accuracy of the assembly, we aligned it to the reference genome using the nucmer tool in the MUMmer program [155] and extracted the percent identity of all one-to-one alignments. The assembly of NA12878 was generated with Oxford Nanopore data

and had very low consensus accuracy of 95.20% when compared to the human reference genome. When this assembly was polished with previously generated Illumina WGS data (SRA: ERP001229), the consensus accuracy was increased to 99.29%. This serves as an upper bound to the maximum accuracy obtained by polishing with both Hi-C libraries. When polished with Prime and Hi-C library, the consensus accuracy was increased to 98.17% and 97.74% respectively. The chicken genome assembly was done with PacBio long read data; thus the consensus accuracy was high (99.86%). When polished with Illumina WGS data, the accuracy was increased to 99.90%. Thus, we observe in this case that polishing does not increase the consensus accuracy by the large amount. When polished with Prime library, the accuracy was 99.89%, very close to what we had obtained by polishing with Illumina library. When polished with a Hi-C library, the consensus accuracy was decreased to 99.84%. Hence, on these two genomes, the Prime library provides more accurate polishing than standard HiC libraries. This difference in the consensus quality can be attributed to the more uniform coverage provided by the Prime library.

Variants	Linking Data	Total Phased Variants	Correctly Phased variants	% Correctly Phased Variants	Switch error
Illumina	Prime	1366821	1361906	99.64%	0.0027
	Hi-C	1177153	1172170	99.57%	0.0033
Prime	Prime	1335292	1330751	99.65%	0.0026
Hi-C	Hi-C	1173218	1179232	99.49%	0.0036

Table 4.3: Accuracy of haplotype phasing using Prime and Hi-C data in NA12878 human genome

4.3.4 Accuracy of haplotype phasing

Next, we assessed the accuracy of haplotype phasing using both the libraries on the NA12878 human genome. We ran HapCUT2 to phase haplotypes and compared them

against the gold standard variant calls for NA12878 human genome released by Genome In A Bottle (GIAB) consortium [165]. We could not do this analysis on the chicken genome because correct phasing information is not available for that genome. Table 4.3 shows the statistics of haplotype phasing using different combinations of variants called with different technologies and linked using Prime and Hi-C data. We observed that the haplotype switch error was lesser while using Prime data than Hi-C. Also, the number of variants phased when the Prime variants used for phasing was comparable to when Illumina variants were used for phasing. This analysis shows that just a single library can be used for both calling variants and linking distant variants to produce accurate phasing.

4.4 Conclusion

In this work, we explored the possibility of having a single sequencing library for scaffolding, polishing, and haplotype phasing large eukaryotic genomes. We used a modified Hi-C library called Prime, developed by Arima Genomics to evaluate the scaffolding, polishing, and phasing accuracy on Human and Chicken genome. We observed that Prime library had fewer inter-chromosomal contacts and more uniform sequencing coverage than the Hi-C library and hence facilitates better scaffolding and polishing accuracy. Also, for the human genome, phasing accuracy was higher with Prime library than Hi-C. Thus, just scaffolding and polishing contig assemblies only with Prime libraries can yield almost gold standard assemblies for large genomes. Because of this, generating Prime libraries along with long read sequencing data can significantly bring down the cost of genome sequencing project.

Our experiments and observations were based on the Prime libraries obtained from only two genomes, whereas Hi-C libraries have been used to scaffold variety of genomes, including various animal and plant genomes. The universal applicability of the Prime library for various aspects of genome assembly is the area of future research, and the data from different species need to be generated for extensive validation. Since the Prime library has fewer inter-chromosomal contacts at the cost of fewer long-range interactions, understanding the effect of this tradeoff on scaffolding contiguity and accuracy in different genomes is the immediate next step. Once the utility of the Prime library for scaffolding is established, a scaffolding method specifically designed to work with this data can be developed to produce even better chromosome-scale assemblies. We tested the accuracy of haplotype phasing only in the human genome which has low heterozygosity (about 0.1%). For the genomes with high heterozygosity, it is unclear how the phasing accuracy would vary when using the Prime library. More validation needs to be performed in order to generalize the use of the Prime library for haplotype phasing to different genomes. Also, HapCUT2, the only phasing method which uses Hi-C data to link distant variants, is specially tuned to the properties of Hi-C data and may yield sub-optimal results with Prime data. Hence, new algorithms for haplotype phasing need to be devised to accommodate the information provided by Prime library. The benefit of Prime data for base level polishing of the assembly is a clear benefit over Hi-C data as Hi-C data has non-uniform sequencing coverage. We believe at this point, Prime data cannot be a drop-in replacement for Hi-C data, but more experimental and computational validation can make a strong case for Prime data to be used routinely in genome sequencing projects.

Chapter 5: Better identification of repeats in metagenomic scaffolding

5.1 Introduction

Genomic repeats are the most critical challenge in genomic assembly even to isolate genomes. When reads are shorter than the repeats (a common situation until the recent development of long read sequencing technologies) it can be shown that the number of genome reconstructions consistent with the read data grows exponentially with the number of repeats [69]. The use of additional information to constrain the one genome reconstruction representing the actual genome being assembled leads to computationally intractable problems. In other words, when reads are shorter than repeats the correct and complete reconstruction of a genome is impossible. In the case of isolate genomes, long read technologies have primarily addressed this challenge, at least for bacteria where the majority of genomic repeats fall within the range of achievable read lengths [166]. In metagenomics, however, the problem is compounded by the fact that microbial mixtures often include multiple closely-related genomes differing in just a few locations. The genomic segments shared by closely related organisms – inter-genomic repeats – are substantially larger than intra-genomic repeats and cannot be fully resolved even if long read data were available. Instead, the best hope is to identify and flag these repeats in order to avoid mis-assemblies that incorrectly span across genomes.

To date, most approaches for repeat detection have been based on the basic observation that repetitive segments have unusual coverage depth, the fact which is usually ascertained through simple statistical tests. These approaches, however, fail in the context of metagenomic data as well as in other settings (e.g., single-cell genomics) that violate the assumption of uniform depth of coverage within the genome, the assumption that is critical for the correctness of statistical tests. Furthermore, the challenges posed by repeats to assembly algorithms are not directly related to the depth of sequencing coverage within contigs; instead, they result from the fact that repeats "tangle" the assembly graph. More specifically, the correct genomic sequence (whether of a single genome or mixture of genomes) can be represented as one or more linear sub-paths of the graph. Repeats induce links within the graph that are inconsistent with this linear structure, making it difficult for algorithms to reconstruct the true genomic structure. We, therefore propose an operational definition of genomic repeats as those nodes in the graph that induce inconsistencies. This definition is orthogonal to the depth of coverage considerations - high coverage contigs that do not "tangle" the graph do not impact assembly algorithms, while contigs that confuse the assembly need to be removed whether or not they can be conclusively labeled as "high coverage."

We have previously proposed an operational definition of repeats regarding betweenness centrality. Bambus 2 [167] scaffolder implemented this approach and is a critical component of the MetAMOS metagenomic assembly pipeline [168]. An example of the effectiveness of this approach in a simple community composed of two genomes is shown in Fig. 5.1. The full implementation of betweenness centrality, however, requires an all-pairs shortest path computation which is computationally too intensive for

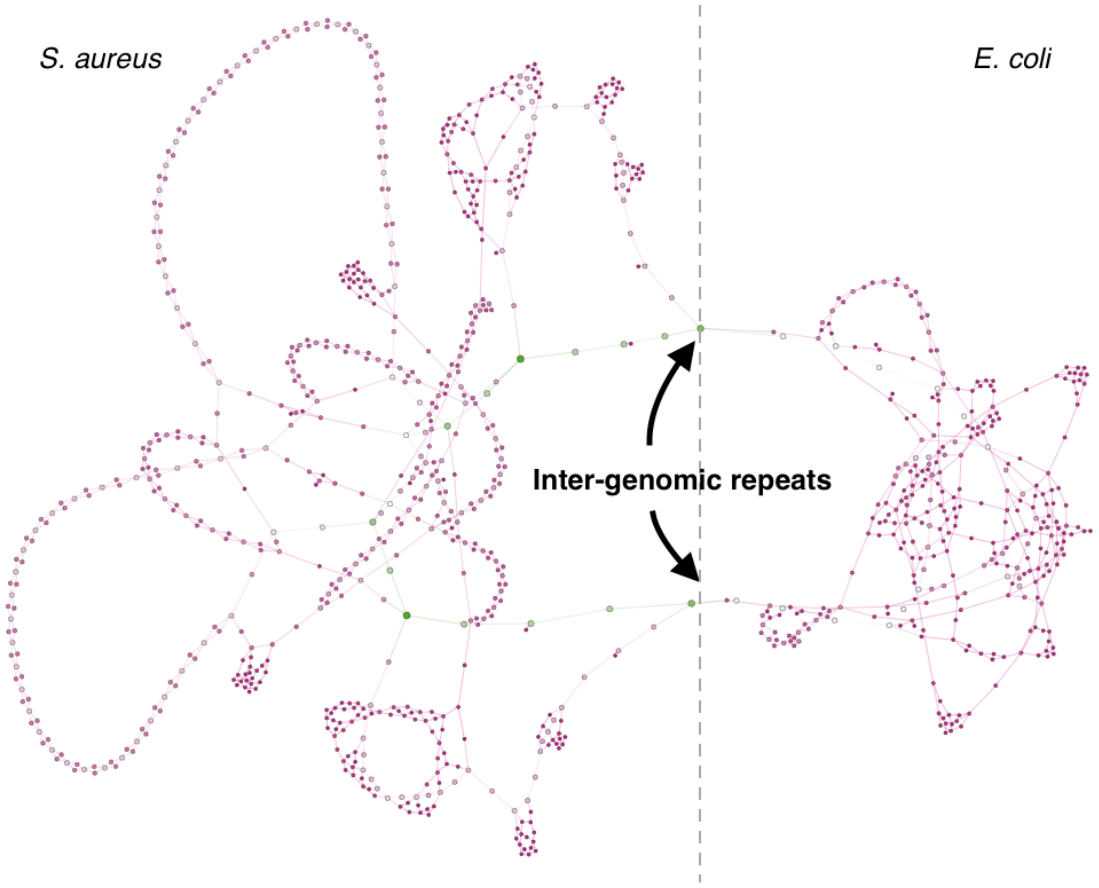


Figure 5.1: Assembly graph of a simulated community consisting of 200 Kbp subsets of *Escherichia coli* str. K-12 MG1655 and *Staphylococcus aureus*. Nodes are colored and sized based on their relative betweenness centrality with larger, green nodes indicating a higher centrality. The highlighted nodes are inter-genomic repeats whose deletion would separate the graph. Note that the betweenness centrality measure correctly identifies these nodes.

typical metagenomic datasets. In Bambus 2, for example, repeat finding in a typical stool sample requires days of computation. To overcome this limitation, we demonstrate here that substantial speed-ups can be obtained through the use of approximate betweenness centrality algorithms without sacrificing accuracy. We further extend this operational definition of repeats by integrating a broader set of graph properties to construct an efficient and accurate repeat detection strategy.

5.2 Related work

5.2.1 Repeat detection in scaffolding

Scaffolding involves using the connectivity information from mate pairs to orient and order pre-assembled contigs obtained from an assembler to reconstruct a genome. This problem of orienting and ordering contigs was shown to be NP-Hard [6]. Various scaffolding methods have been designed based on different heuristics to obtain approximate solutions to the problem. However, all of these methods face difficulties when dealing with contigs originating from repetitive regions in the genome. A common strategy for handling repeats is to identify and remove them from the graph before the scaffolding process, then re-introduce them after the contigs have been properly ordered and oriented. Most of the existing scaffolders use depth of coverage information to classify a contig as a repeat. For example, Opera [169] and SOPRA [60] filter out as repetitive contigs with coverage 1.5 and 2.5 times more than average coverage, respectively. The MIP scaffolder [59] uses high coverage (greater than 2.5 times average) as well as a high degree (≥ 50) of nodes within the scaffold graph to determine repeats. Bambus 2 [167] – a scaffolder specifically designed for metagenomic data – uses a notion of betweenness centrality [170] along with global coverage information to find out repeats.

5.2.2 Betweenness centrality

In network analysis, metrics of centrality are used to identify the most important nodes within a graph. Several metrics to measure centrality have been proposed, but in

this work, we use betweenness centrality. The betweenness centrality of a particular node is equal to the number of shortest paths from all nodes to all others that pass through that node. Intuitively, a node that is frequently found on paths connecting other nodes is a potential repeat, as along a simple path all nodes should have roughly the same centrality value. The algorithm for computing exact centrality [170] takes $\Theta(mn)$ time on a graph with m nodes and n edges. Several solutions were proposed to overcome this computational cost of computing network centrality, including an exact massively parallel implementation [171], and an approximate solution based on sampling a subset of the nodes [172]. Recently, a better parallel approximation algorithm was proposed by Riondato et al. [173] which uses a strategy for sampling from among the shortest paths in the graph to compute betweenness centrality. The size of the chosen sample of paths can provide provable bounds on the accuracy of the centrality value given by the algorithm. The sample size is determined as a function of an approximation factor ϵ and the diameter of the graph.

5.3 Methods

5.3.1 Construction of scaffold graph

A scaffold graph is defined as a graph $G(V, E)$, where V is set of all the contigs. The edges represent links between the contigs inferred from read pairing information – if the opposite ends of a read pair map to different contigs we can infer the possible adjacency of these contigs within the genome. Since most genome assemblers do not report the location of reads within contigs, we infer this information by mapping using

bowtie2 [77]. Experimental library size estimates are often incorrect, and we re-estimate here the distance between the paired reads from pairs of reads mapped to the same contig. We record the average insert size l and standard deviation $\sigma(l)$ within a library. For each pair of contigs, we retain the maximal set of links that are consistent regarding the implied distance between the contigs for each implied relative placement of the contigs. Since contigs can be oriented in forward or reverse direction depending on the orientation implied by mapped mate pairs, there exist four possible orientations of adjacent contigs (forward-forward, forward-reverse, reverse-forward and reverse-reverse). For each of the possible relative orientation, we need to find a maximal set of consistent links implying that orientation. This set can be identified in $O(nlogn)$ time using an algorithm to find maximal clique in an interval graph [62]. The distance between the contigs implied by the resulting "bundle" of links has mean $l(e) = \frac{\sum \frac{l}{\sigma(l)}}{\sum \frac{1}{\sigma(l)^2}}$ and standard deviation $\sigma(l) = \frac{1}{\frac{1}{\sigma(l)^2}}$, as suggested by Huson et al. [58].

5.3.2 Orienting the bidirected scaffold graph

The scaffold graph derived from the process outlined above is bidirected [174]. It can be converted into a directed graph by assigning an orientation to each node, reflecting the strand of the DNA molecule that is represented by the corresponding contig. In computational terms, we need to embed a bipartite graph (the two sets corresponding to the two strands of DNA being reconstructed) within the scaffold graph. In the general case, such an embedding is not possible without removing edges in order to break all odd-length cycles in the original graph. Finding such a minimum set of edges is NP-

Hard [175]. We use here a greedy heuristic proposed by Kececioglu and Myers [6] which achieves a $2-$approximation and runs in $O(V + E)$ time.

5.3.3 Repeat detection through betweenness centrality

We start by calculating centrality values for all the nodes in the graph using either an exact or approximate centrality algorithm as outlined in the introduction. Let μ be the mean and σ be the standard deviation of the resulting centrality values. A contig is marked as a repeat if its centrality value is greater than $\mu + 3 * \sigma$. This cutoff criterion is the same as the one used in Bambus 2. We have also experimented with other definitions of outliers (such as interquartile range); however, the original definition used in Bambus 2 performed better than the interquartile range cutoff (data not shown).

5.3.4 Repeat detection with an expanded feature set

Centrality is just one of the possible signatures that a node in the graph "tangles" the graph structure, making it harder to identify a correct genomic reconstruction. At a high level, one can view centrality to relate to difficulties in ordering genomic contigs along a chromosome. The orientation procedure outlined above provides potential insights into contigs that may prevent the correct orientation of contigs – contigs adjacent to a large number of edges invalidated by the orientation procedure are possible repeats. Other potential signatures we consider include the degree of graph nodes (highly connected nodes are potential repeats) as well as abrupt changes in coverage between adjacent nodes. The latter information is defined as follows. For each contig, we capture the distribution

of read coverage values. We then use a Kolmogorov-Smirnov test [176] to identify pairs of contigs that have statistically different distributions of coverage values. We flag all edges that exceed a pre-defined p-value cutoff (in the results presented here we use 0.05). We combine these different measures (contig length, centrality, node degree, a fraction of the number of edges invalidated by the orientation routine that are adjacent to a node, fraction of number of edges with abrupt changes in coverage, and the ratio of contig coverage to average coverage) within a Random Forest classifier [177].

To generate training information for the classifier, we aligned the contigs to an appropriate set of reference genomes using MUMmer [92] dependent on the data being assembled, and flagged as repetitive all contigs that had more than one match with greater than 95% identity over 90% of the length within the reference collection.

5.4 Results

5.4.1 Dataset and assembly

To test our methods, we used a synthetic metagenomic community dataset (S1) by Shakya et al. [178] that was derived from a mixture of cells from 83 organisms with known genomes. Reads in the datasets were cleaned and trimmed using Sickle [179]. Assembly was performed using IDBA-UD [66] with default parameters. The assembly of S1 yielded 47,767 contigs.

5.4.2 Extended feature set improves repeat detection

We trained a Random Forest classifier that takes into account the various measures outlined above as follows. We simulated a low coverage (10x) dataset using a read simulator provided with the IDBA assembler from the set of 40 genomes downloaded from NCBI [1]. We constructed contigs from the simulated reads and mapped them to reference sequences to identify which contigs are repetitive (have ambiguous placement in the reference set). We used this information to train the classifier, then used the resulting classifier to predict repeats within the synthetic community S1 described above. As can be seen in Figure 5.2 the accuracy of the classifier based on multiple graph properties is higher than that of approaches that rely on just coverage as a criterion to classify a contig as a repeat. Classification of repeats using approximate centrality provides higher specificity compared to the coverage approach at the cost of slightly lower sensitivity. The Random Forest approach leverages the advantage of high sensitivity from the coverage approach and high specificity from the centrality approach along with some additional features to provide better overall classification.

5.4.3 Important parameters in determining repeats

We further explored the features of the data that contribute to the better performance of the classifier. In Figure 5.3 we show the contribution of each feature to the classifier. The length of contigs, a factor not usually taken into account when detecting repeats, appears to have the most significant influence. This is perhaps unsurprising as

[1] ftp://ftp.ncbi.nlm.nih.gov/genomes/bacteria/all.fna.tar.gz

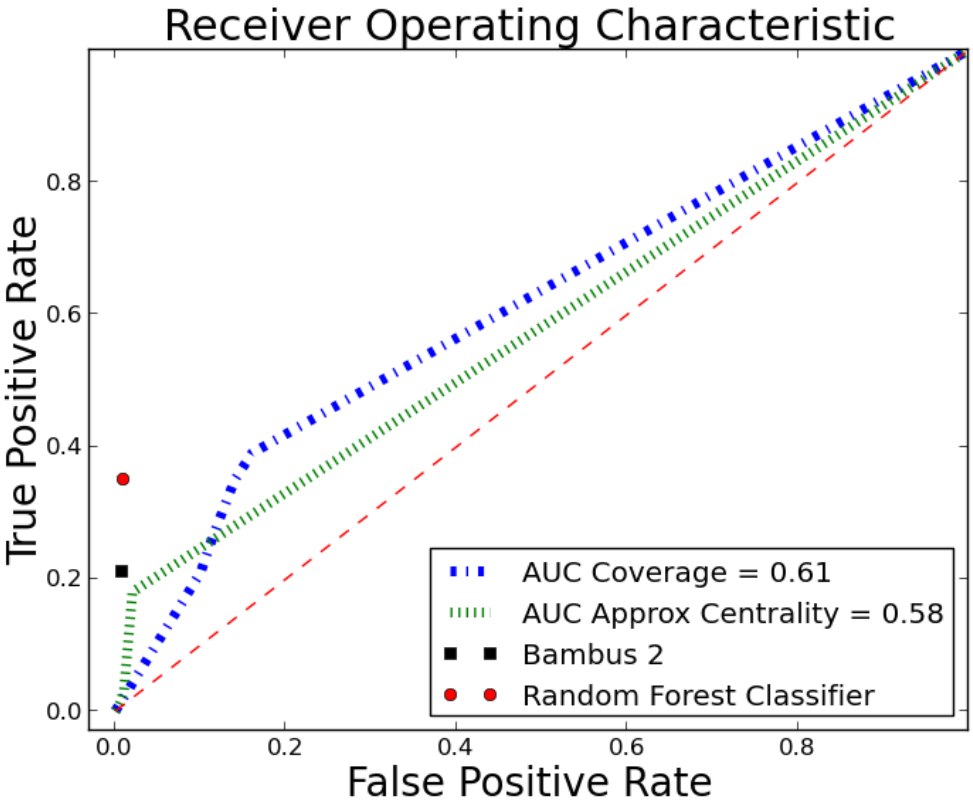


Figure 5.2: Plot for comparison of Random Forest classifier with the coverage and centrality approach. The red circle in the plot indicates the sensitivity and specificity obtained by using the Random Forest approach. The black square in the plot indicates the sensitivity and specificity obtained by using Bambus 2.

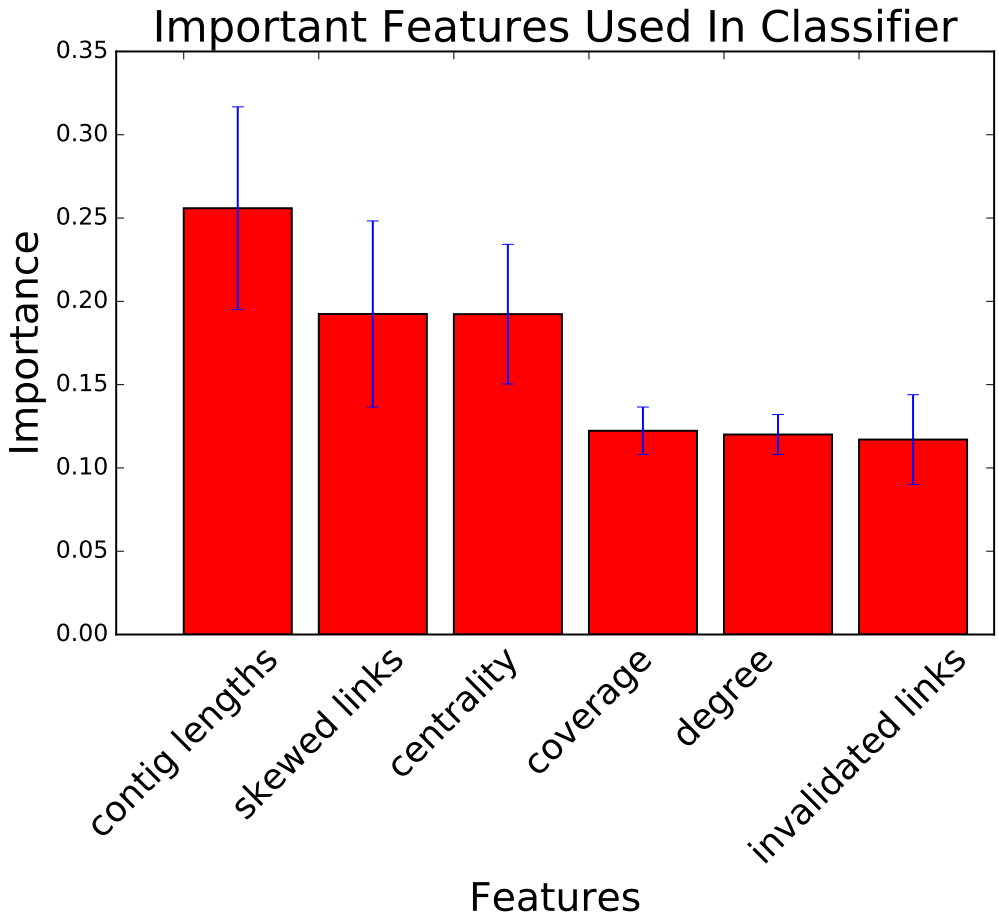


Figure 5.3: Importance of features used in Building Random Forest classifier

repeats confuse the assembly process as well, fragmenting the assembly. In other words, longer contigs are less likely to represent repetitive sequences. The second most essential features are the fraction of edges adjacent to a contig that indicate an abrupt change in coverage. Contigs with unusual coverage in comparison to their neighbors can also be reasonably assumed to be repetitive. Centrality was the third most important factor, as expected. Perhaps surprising, the overall depth of coverage or node degree are not as important as features despite these measures being among the most widely used signatures of "repetitiveness" by existing tools.

5.4.4 Comparison of incorrectly oriented pair of contigs

Beyond testing the simple classification power of different approaches, we also evaluated the different methods regarding whether the removal of nodes marked as repeats makes the scaffolding process more accurate. Specifically, we explored how different repeat removal strategies affect the contig orientation process. The scaffold graph for the S1 dataset had 21,950 nodes and 31,059 edges. We removed the repeats reported by the different methods from this graph and oriented the resulting graph. We then tracked the accuracy of the results regarding the number of edges that imply a different relative orientation of the adjacent nodes than the correct one, inferred by mapping the contigs to the reference genomes. Here the relative orientation can either be same if both the contigs on edge have the same orientation (forward-forward and reverse-reverse) and different if the contigs on edge different orientations (forward-reverse and reverse-forward). The results are shown in Table 5.1. The centrality-based methods and the Random Forest classifier based methods resulted in lower error rates and retained a higher percentage of the edges in the original graph than coverage based methods.

5.4.5 Comparison of runtime with bambus 2

The results above show that Bambus 2 has, unsurprisingly, a similar level of accuracy with the approximate centrality approach. We have already mentioned, however, that Bambus 2 is inefficient on large datasets. To explore the efficiency of the approximate centrality approach, we used a real metagenomic dataset (SRX024329 from NCBI) representing sequencing data from the tongue dorsum of a female patient. Assembly of

Method	**Correct**	**Wrong**	**% Correct**	**% Wrong**
Bambus 2	12042	867	38.77 %	4.11%
Approximate Betweenness Centrality	12336	917	39.71%	3.94%
Coverage (MIP, SOPRA)	3840	315	17.49%	4.72%
Coverage (Opera)	2007	165	6.46%	5.62%
Random Forest	12255	807	39.45%	3.52 %

Table 5.1: Number of correctly and incorrectly oriented links in scaffold graph using various repeat removal strategies. The % correct column represents the percentage of correctly oriented links as a function of the total number of edges in the original scaffold graph. % wrong column represents the percentage of incorrectly oriented links in the graph obtained by removing repeats.

these reads was performed using IDBA yielding 106,380 contigs in total. The scaffold graph constructed from these contigs had 112,502 edges. The 'MarkRepeats' module of Bambus 2 took almost 2 hours to detect repeats, whereas the approximate betweenness centrality algorithm found repeats in approximately 5 minutes, a substantial improvement in speed without a loss of accuracy as shown above. To compare the runtime with training of Random Forest classifier, we trained the classifier on contigs in this dataset. Since we did not have reference sequences for this dataset, we randomly marked a subset of contigs as repeats and performed training. It took about 20 minutes to calculate features and fit a classifier which was still faster than the time taken by Bambus 2.

5.5 Discussion and conclusion

Our prior work had introduced the use of network centrality as an approach for detecting repeats in metagenomic assembly, a setting where coverage-based approaches are often ineffective. This approach, implemented in the scaffolder Bambus 2, was, however, inefficient for large datasets, the fact that has limited its use. Here we extend our original

approach by incorporating multiple features of the scaffold graph (including centrality) that may be signatures of repetitive sequences within a Random Forest classifier. We also show that an approximate calculation of network centrality based on the random sampling of paths obtains similar accuracy as the full centrality computation at a fraction of computational time.

Our results demonstrate that methods that directly capture the effect of repeats on the assembly graph are more effective at detecting repeats than indirect measures such as depth of coverage, particularly in the context of the metagenomic assembly. Our new approach improves in both accuracy and efficiency over existing methods for repeat detection, and we plan to incorporate it within the MetAMOS metagenomic assembly pipeline as a replacement for the existing code within Bambus 2. We note that the classification accuracy was surprisingly high even though the classifier was trained on purely simulated data yet applied to the real dataset. This validation underscores the robustness of the feature set we have identified. At the same time, the graph features that we have identified as useful in detecting repeats are just a first step towards a better understanding of the features of the data that most influence the ability of assembly algorithms to accurately reconstruct metagenomic sequences. Also, classifiers like Random Forest can be implemented in parallel [180] which can provide significant runtime speedups for large metagenomic datasets. We plan in future work to further explore both the feature set and the approaches used to build and train the classifier to increase accuracy and ultimately improve the quality of metagenomic reconstructions.

Chapter 6: MetaCarvel: linking assembly graph motifs to biological variants

6.1 Introduction

Sequencing of DNA directly extracted from microbial communities (metagenomics), has emerged as a key tool in the exploration of the role microbes play in human and environmental health. Large-scale studies enabled by metagenomic methods, such as MetaHIT [36] and the Human Microbiome Project (HMP) [181] have cataloged the complex microbial communities associated with the human body and have demonstrated their importance to human health. By eliminating the need for culturing, metagenomic sequencing has made it possible to explore a broader range of the microbes inhabiting our world and has led to the discovery of novel organisms and genes from complex samples [182–185].

Despite promising initial results, the reconstruction of entire or even partial organisms from complex microbial mixtures remains a tremendous challenge. The assembly of metagenomic sequences is confounded by several factors: (i) uneven abundance of the different organisms found in a sample, (ii) genomic variation between closely related organisms, (iii) conserved genomic regions shared by distantly related genomes

(inter-genomic repeats), and (iv) repetitive DNA within a single genome (intra-genomic repeats). All but the latter challenges are unique to metagenomic data and have not been the target of research until very recently.

Several genome assembly tools designed explicitly for metagenomic data have been developed in recent years. Among the most widely used are metaSPAdes [67] and MEGAHIT [186], however many other tools have been developed including MetaVelvet [187], IDBA-UD [66], Ray Meta [188], and Omega [189]. These tools effectively address the uneven coverage of metagenomic datasets, but virtually all of them "smooth out" small differences between co-occurring strains of organisms in order to enable the reconstruction of longer genomic segments from the mixture. Furthermore, the output of the assemblers is simply a collection of linear segments (contigs) that lacks the connection between the segments originating from the same organism. As a result, additional analyses are necessary to discover information about the adjacency of genomic segments (e.g., operon structure in bacteria), or large-scale genomic variants between co-occurring microbial strains. The latter information is of particular research interest in microbial ecology, for example in the context of the lateral gene transfer [190] or understanding how genomic heterogeneity contributes to the stability of microbial communities [191].

The study of genomic variants in microbial communities is of considerable interest, and a number of computational tools have been developed to discover this information. The approaches are primarily based on read alignments to either complete genomes, as performed for example by metaSNV [192] and MIDAS [193], or against conserved genes, as performed by ConStrain [194] and StrainPhlan [195]. Strain variants can also be discovered directly from the output of the assembler, as done, for example, for diploid

genomes through a colored de Bruijn graph approach [196], or in metagenomic data through the use of the SPQR tree data structure [197].

The discovery of genomic variants from assembly relies on the information contained in an assembly graph - a representation of the ambiguity in the reconstruction of the genome or metagenome. While many assemblers can output this information, an assembly graph can also be constructed post-assembly by linking together genomic contigs through the information provided by paired reads or other sources of information, using a computational process called scaffolding. While most existing genome and metagenome assemblers [12, 67, 198] contain dedicated scaffolding [134] modules, the output of these tools comprises linear paths that ignore the presence of genomic variants. An exception are stand-alone scaffolders such as Bambus 2 [167] or Marygold [197] that explicitly retain ambiguity in the assembly graph and use graph analyses to characterize specific genome variants.

Here we describe a new metagenomic scaffolding package called MetaCarvel (Figure 6.1), a tool that substantially improves upon the algorithms implemented in Bambus 2 and MaryGold. We show that MetaCarvel generates more contiguous and accurate scaffolds than one of the best performing stand-alone scaffolders (OPERA-LG) [64, 199], and that it can accurately detect a number of genomic variants, including regions with divergent sequence, insertion-deletion events, and interspersed repeats. MetaCarvel is released under MIT open source license and is available at https://github.com/marbl/MetaCarvel.

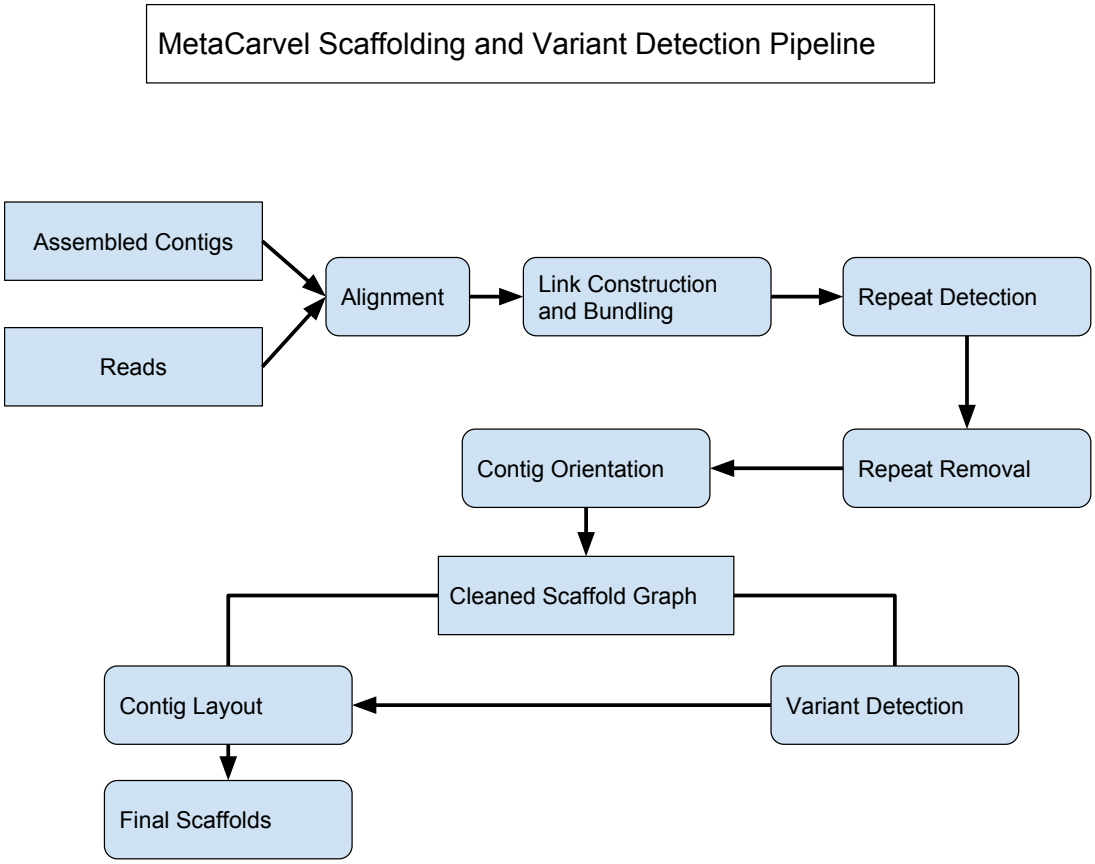


Figure 6.1: Overview of MetaCarvel pipeline: First, reads are aligned to assembled contigs. Using these alignments, a scaffold graph is constructed by bundling the link implying same contig orientation. In this graph, repeats are identified and removed. After the graph is cleaned, variants are identified and collapsed. In this simplified graph, the scaffolds are generated by traversing linear paths.

6.2 Methods

6.2.1 Contig graph construction

MetaCarvel begins by aligning paired-end reads to assembled contigs using a standard short read aligner such as BWA [78] or bowtie2 [77]. The reads are aligned in single end mode to avoid biasing alignments based on the library insert size specifications. Using the alignments of reads to contigs, a contig graph is created where the nodes are contig

sequence and edges are the linking information provided by the paired-end reads where each read in the pair is aligned to different contigs. We first re-estimate the library size (mean and standard deviation) by considering a set reads where both reads in the pair are aligned to the same contig. For a pair of contigs, different read pairs can imply vastly different distance between those contigs. To avoid this issue, we first compute the maximal set of links that are consistent to each other and imply similar distance (anywhere between $\mu - 3 * \sigma$ and $\mu + 3 * \sigma$, where μ and σ are the mean and standard deviation of the input paired-end library) between two contigs. Finding such set of consistent links is similar to finding a maximal clique in an interval graph. In this case, intervals can be imagined as the read pair and the position of that interval is dictated by the alignment coordinates. With this formulation, the maximal set of consistent links can be computed in $O(NLog(N))$ time, where N is the number of total read pairs aligned to contigs [62]. Once the set of mutually consistent links is identified, they are "bundled" into a single representative link. The mean and standard deviation for this link is computed using a method described in Huson et al. [58]. The weight of this link is given by the number of link which were bundled while constructing the link.

6.2.2 Repeat detection

To avoid the ambiguities caused by genomic repeats during scaffolding, we first identify repetitive contigs and remove them from the contig graph along with all the edges incident on them. We use different signatures for finding repetitive contigs that could confound the scaffolding process [200]. First, we calculate the sequencing coverage and

degree for all the contigs in the graph. Then, we assign a unique orientation to each contig in the contig graph using an algorithm described in more detail in the next section. This algorithm removes edges from the graph that prevent the assignment of a consistent orientation to contigs. For each contig, we count the number of edges such removed. We also flag links in the contig graph that connect contigs with significantly different depths of coverage. We track how many such "skewed" links are incident on each contig.

For each of the features described above (depth of coverage, node degree, incident edges invalidated during the orientation phase, skewed edges), we flag the contigs that occur within the upper quartile of all contigs. Any contig that is flagged according to most of (at least 3) the criteria listed above is marked as a repeat and removed. After removing these contigs, we also remove contigs with a high betweenness centrality [170] measure (the number of shortest paths passing through a node in a graph) - specifically the contigs that have a betweenness centrality higher by more than 3 standard deviations from the mean betweenness centrality for the assembly graph. Since the computation of betweenness centrality is computationally expensive ($O(N^3)$ for N contigs), we use an approximation algorithm [173] which runs in linear time.

6.2.3 Orientation

The contig graph is bidirected because each contig in the graph can originate from either forward or reverse DNA strand. To make this graph directed, we need to assign a unique orientation to each contig. The edges in the graph are of two types: "same" when adjacent contigs have same orientation and "different", otherwise. If the graph

has an odd length cycle of "different" edges, then it is impossible to assign a consistent orientation to contigs in that cycle without discarding at least one edge from the cycle. Our objective is to minimize the number of edges to be removed from the graph in order to allow a consistent orientation for all contigs. Finding such minimum set is equivalent to finding a maximal bipartite subgraph - an NP-Hard problem [175]. We use the greedy algorithm described in Kelecioglu et al. [6] that achieves a 2-factor approximation and runs in $O(V + E)$ time (V - the number of contigs, E - the number of edges connecting these contigs). Briefly, we assign an arbitrary orientation (forward or reverse) to a starting contig, then proceed to orient all contigs adjacent to it. While assigning an orientation to a contig, we pick an orientation in such a way that it agrees with the majority of its already oriented neighbors (in terms of edge weights supporting that orientation). Once we assign an orientation to a contig, we invalidate any links that disagree with the chosen orientation. We continue in a breadth-first manner and assign an orientation to all the contigs.

6.2.4 Bubble collapsing

A typical metagenomic sample contains closely related genomes or closely related strains of same organism which result in complex bubble-like pattern in the graph. Identifying complex bubbles in the graph takes exponential time in the number of nodes, thereby making bubble identification extremely slow on large and complex metagenomics samples. To identify bubbles in the graph efficiently, we first decompose the oriented contig graph into its biconnected components using Hopcroft-Tarjan algorithm [201]. This takes

$O(|V|+|E|)$ time. We further decompose each biconnected component into triconnected components by computing an SPQR tree data structures [202,203]. SPQR tree for a graph denotes a hierarchical decomposition of biconnected components of a graph into its triconnected components and each tree node corresponds to a triconnected component in the original graph. Using this tree, we extract separation pairs (a pair of nodes when removed from a connected component make it disconnected) in the graph. We used the implementation of SPQR trees provided in Open Graph Drawing Framework (OGDF) [204] which runs in linear time $O(|V| + |E|)$. We then test if a separation pair is a valid source-sink pair in the assembly graph so that it can span a bubble using an algorithm used in Marygold. Briefly, we check if all the paths starting at the source of the bubble end at the sink of that bubble and invalidate all the separation pair for which this condition does not hold. Once valid source-sink pairs and bubbles are identified, each bubble is collapsed into a supernode. The incoming and outgoing edges from the source and sink respectively for the bubbles are assigned to its supernode. This simplifies the graph structure by large extent thereby masking the complexities caused by the variants in the sample

The graph components we identify are also reported by MetaCarvel as putative strain variants, allowing further analysis. From among the patterns identified we have focused the analysis in this paper on three simple patterns (refer to Figure 6.2).

Three bubbles: Three node bubbles in the graph correspond to putative gene gain-loss events in the genome, hence, are important from the biological point of view. These bubbles can be easily found from the validated bubbles of size 3.

Four bubbles: Four node bubbles correspond to putative variation between the genomes of related strains within a sample. Like three bubbles, they can also be easily character-

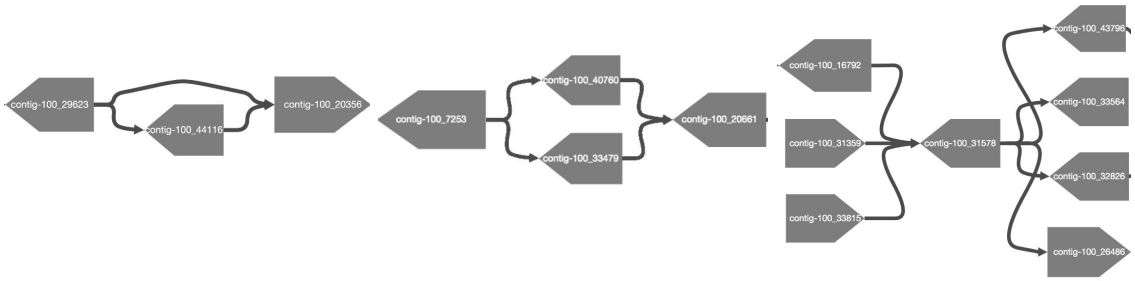


Figure 6.2: Different types of motifs detected by MetaCarvel. (A) Three node bubbles potentially represent gene gain/loss events and horizontal gene transfers. They are formed due to the insertion or deletion of chunks between two otherwise similar genomes. (B) Four node bubbles denote the variation between very similar sequences. They can result in the graph due to the species with very high sequence similarity. (C) Interspersed repeats in the graph are denoted by the nodes with high centrality and usually tangle the graph.

ized within the validated bubbles obtained during the bubble collapsing step.

Interspersed Repeats: Interspersed repeats are natively detected and flagged by the repeat detection procedure described above.

6.2.5 Linear scaffold generation

Once we simplify the graph by collapsing bubbles into supernodes, we generate the scaffold sequences through a linear traversal of the graph. We first create an auxiliary graph $G'(V', E')$ from the original graph $G(V, E)$, as follows. We create two nodes for each contig, one for the $5'$ end and one for the $3'$ end, connected by an edge that matches the orientation of the corresponding contig. The edge weights for E' is the bundle sizes (number of mate pairs supporting that edge). The edges between the $5'$ and $3'$ ends of same contigs are not added at this stage. We then compute a weighted maximal matching [205] in G'. After we compute a weighted maximal matching, we remove nodes and edges present in that matching and repeat the matching process on the remaining nodes and

edges until all nodes in G' are matched. In each maximal matching, we add edges between the $5'$ and $3'$ ends of each contig present in that matching. This defines a unique linear path in G' and spells out a scaffold. We note that supernodes (collapsed regions of strain variation) can be part of the linear path constructed from the scaffold graph. Since each bubble is a directed acyclic graph (DAG), we compute the highest weighted path from source to sink within each supernode using a dynamic programming algorithm. This path is then merged within the global linear path to define the linearized scaffold. For each supernode, we also output additional source to sink paths as alternate variants by iteratively removing edges that were previously reported.

6.3 Results

6.3.1 Effect of microbial mixtures on scaffolding

We compared the performance of MetaCarvel to that of OPERA-LG [64], using both single genomes and an increasingly complex mixture of genomes. We used reads from five different strains of *Acinetobacter baumanii* (NCBI Short Read Archive accessions SRR1008889, SRR1030406, SRR1019232, SRR1030403, and SRR1030473) and assembled them using both MEGAHIT and metaSPAdes. To simulate the impact on scaffolding performance of increasing levels of genome heterogeneity among closely related organisms, we created increasingly complex mixtures comprising from one to five genomes. We aligned the paired reads to the resulting assemblies and used MetaCarvel and OPERA-LG to perform scaffolding. As more genomes are added to the mixture the quality of the assembly degrades and so does the quality of the resulting scaffolds

for MEGAHIT assembly (Figure 6.3(A)). However for metaSPAdes contigs and resulting scaffolds, the contiguity increases as more genomes are added to the mixture because metaSPAdes aggressively tries to assemble genomes without preserving much variation in the assembly (Figure 6.3(B)). Even in the case of the assembly of a single genome, scaffolding with MetaCarvel improves contiguity, albeit by only a small amount (13.31 kbp contig NG50 vs.18.51 kbp scaffold NG50 using MEGAHIT and 16.96 kbp contig NG50 vs. 18.99 kbp scaffold NG50 using metaSPAdes). The contiguity (sum of lengths of contigs on a linear path in the graph) of the scaffolds generated by MetaCarvel substantially improves over the original assembly for the more complex samples. When compared to metaSPAdes scaffolds (generated using the scaffolding module built within this assembler), MetaCarvel's scaffold contiguity was at least as good as metaSPAdes scaffolds for all mixtures (Figure 6.3(B)). The contiguity of the scaffolds degrades slower than that of the scaffolds generated by OPERA-LG even as the contiguity of the underlying contigs created by MEGAHIT and metaSPAdes degrades rapidly as the complexity of the mixture increases.

To measure the correctness of the assemblies, we computed the number of mate-pairs mapped concordantly, that is, the mate pairs whose two ends are properly oriented with respect to each other and the distance between the paired reads is within the insert size limit implied by the library. This measure is correlated with assembly quality as mis-assemblies, or fragmented contigs and scaffolds result in unmapped reads and discordant mate-pairs. For all the mixtures and both assemblers, MetaCarvel scaffolds had the highest number of concordant mate pairs (Figure 6.3(D) and (E)).

As the number of genomes in a mixture increased so did the number of genomic

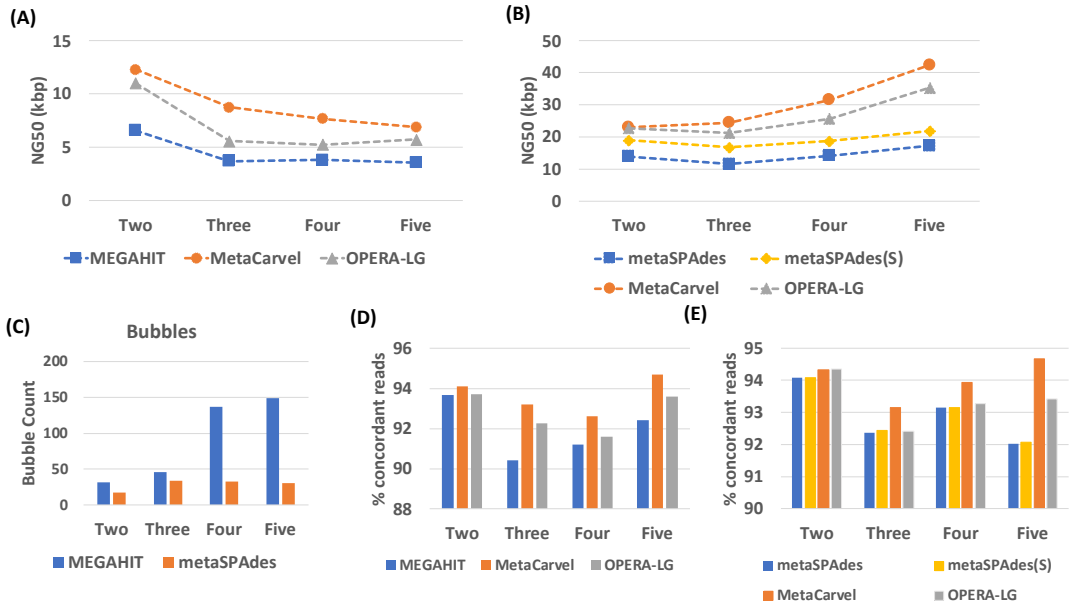


Figure 6.3: Scaffold statistics for *Acinetobacter baumannii* strain mixtures:(A) NG50 statistics when MEGAHIT contig assembly was used as an input for scaffolding methods. (B) NG50 statistics when metaSPAdes contig assembly was used as in input for scaffolding methods. metaSPAdes(S) denotes the scaffolds generated by inbuilt scaffolding module of metaSPAdes. (C) Number of bubbles detected by MetaCarvel for different input assemblies. The percentage of reads aligned concordantly when (D) MEGAHIT assembly was used as an input and when (E) metaSPAdes assembly was used as in input.

variants detected by MetaCarvel (Figure 6.3(C)). A sample pattern is shown in Figure 6.4. In this example, the parallel contigs differed by about 3% nucleotide identity, a value larger than the amount of error tolerated by the assemblers. We observed that the number of variants/bubbles detected by MetaCarvel was much higher with MEGAHIT assemblies compared to metaSPAdes. However, the contiguity of scaffolds generated with metaSPAdes was much higher than the scaffolds generated with MEGAHIT contigs. This highlights the fact that preserving variation in the assembly affects the contiguity. If assembler doesn't smooth out the variations in the genome, it would provide much contiguous assembly at the cost of hidden variation.

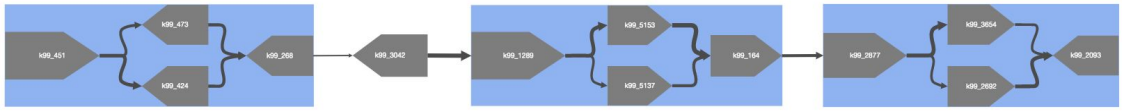


Figure 6.4: Variants detected in one of the components of *Acinetobacter baumanii* scaffold graph. In this component, we find all the non-terminal nodes in a bubble are more than 97% identical to each other and originate from two different strains of *Acinetobacter baumannii* genome.

6.3.2 Accuracy of detection insertions and deletions

To verify the accuracy of detecting insertion and deletions, we used MEGAHIT to co-assemble reads from two strains of *Escherichia coli* for which fully complete reference sequences are available: *Escherichia coli* K12 (NCBI sequence read archive accession: ERR022075) and *Escherichia coli* O83:H1 (NCBI sequence read archive accession: SRR6512538). We scaffolded the resulting assembly using MetaCarvel and flagged as predicted insertion/deletion event the three node bubbles (see Figure 6.2(A)) found within the resulting scaffolds. To flag insertion and deletion events between these two *Escherichia coli* genomes, we aligned them with each other using NUCmer and extracted the regions flagged as "GAP" by the dnadiff utility from the MUMmer package [155]. We determined that a three node bubble represented a true insertion/deletion event if the middle contig aligned within one of these regions. Of 126 three-node bubbles detected by MetaCarvel, 81 were found concordant with the insertion/deletion events identified by MUMmer (64.28% precision). A total of 194 contigs aligned to gap regions within the *E. coli* genomes, implying a specificity of 73.14%. Some of the false negatives (32) were due to the parameters used in MetaCarvel to eliminate low quality graph, while 12 other false negatives were due to the fact that the insertion/deletion event coincided with other

genomic phenomena, leading to a graph motif that was inconsistent with our definition of a three node bubble.

6.3.3 Detection of regions with high sequence variation

To evaluate the accuracy of sequence variants detected by MetaCarvel, we used reads from two fairly distant strains of *Acinetobacter baumannii* genome (SRR1171982 and SRR1200567) [206]. We selected *Acinetobacter baumannii* for this analysis because the variation between different strains is associated to their antibiotic resistance. We co-assembled the reads with MEGAHIT and ran MetaCarvel's variant detection on the resulting assembly. We aligned the contigs to the *Acinetobacter baumannii* 1656-2 reference genome sequence (NCBI ID: NC_017162). The contigs which aligned at a same position in the reference genome were likely to have originated from the true variants (Figure 6.2(B)). MetaCarvel detected 191 variants in this graph, among which 184 overlapped with variants identified by alignment to the reference genome. In the remaining 7 variants which could not be validated using the strain 1656-2, the contigs from these variants were perfectly aligned to *Acinetobacter baumannii* strain AR_0078, *Acinetobacter baumannii* strain XH731, *Acinetobacter baumannii* strain 15A34. For the remaining bubbles, the contigs in those bubbles did not align to any known strain of *Acinetobacter baumannii* with high identity, suggesting possible misassemblies. We also performed a similar analysis on a mixture of *Escherichia coli* K12 and *Escherichia coli* O83:H1 genomes. In this case, to flag a true variation, we check if contigs in a bubble are aligned to both the strains with high identity over at least 95% of their length. With this defini-

tion, 28 of 31 bubbles found by MetaCarvel matched actual variants, implying 90.32% precision.

Dataset	Method	True Repeats	Predicted Repeats	True Positives	False Positives	True Negatives	False Negatives	Sensitivity	Specificity
Yersinia pestis	OPERA-LG	46	353	31	322	904	15	67.39%	73.79%
	MetaCarvel	46	222	33	189	1037	13	71.73%	84.58%
JGI Mock	OPERA-LG	532	771	356	415	21,215	176	66.91%	98.01%
	MetaCarvel	532	454	438	16	21,614	94	82.33%	99.92%

Table 6.1: Comparison of the accuracy of repeat detection in MetaCarvel and OPERA-LG on different datasets.

6.3.4 Effectiveness in detecting repeats

To determine the accuracy of interspersed repeat detection (Figure 6.2(C)), we used reads from *Yersinia pestis* CO92 genome (Genebank ID: AL590842.1) as this genome has well characterized interspersed repeats [207]. We assembled the reads (SRR069183) using MEGAHIT and then scaffolded the assembly with MetaCarvel. To define a ground truth, we aligned the contigs to the *Yersinia pestis* genome using NUCmer (with –maxmatch option) [155] and flagged as repeats all contigs aligned at more than one location with at least 95% identity and 95% alignment length. The sensitivity and specificity of MetaCarvel's repeat detection algorithm was 71.86% and 83.19% respectively. We compare this result to the algorithm used in OPERA-LG which detects repeats using sequence coverage alone (contigs with 1.5 times the average coverage of the genome are flagged as repeats). Within the same assembly of *Yersinia pestis*, OPERA-LG's repeat finding approach has sensitivity and specificity of 67.39% and 73.79%, respectively (Table 6.1). Further, we assessed MetaCarvel's repeat detection accuracy on a synthetic metagenomic dataset described in Singer et al. [208]. We assembled the reads using MEGAHIT, and the resulting contigs were aligned to the reference genomes using NUCmer (with –maxmatch option). In

this case the sensitivity and specificity of MetaCarvel's repeat detection was 82.33% and 99.93% compared to 66.91% and 98.01% of OPERA-LG (Table 6.1). Overall, OPERA-LG found fewer repeats than MetaCarvel, incurring higher number of false positives. The repeats missed by MetaCarvel had inconsistent read alignments and hence were not part of the scaffold graph. Of the 16 false positives obtained from MetaCarvel, 8 of them were marked with 'high coverage node' as one of the features and 3 of them were marked based on high betweenness centrality.

Assembler	Method	Scaffolds	Total Assembly Size	% reference covered	Misassemblies	Largest Scaffold (bp)	Length at 1 Mbp	Length at 10 Mbp	Length at 50 Mbp
MEGAHIT	OPERA-LG	22,597	87,540,947	87.19%	207	47,3251	319,244	97,007	2,340
	MetaCarvel	19,879	88,092,845	88.09%	99	287,352	287,352	154,353	9,127
metaSPAdes	OPERA-LG	4,931	98,714,003	91.25%	148	2,331,214	2,055,240	944,070	177,154
	metaSPAdes(S)	5,333	98,302,677	91.12%	55	1,991,282	1,810,249	158,300	199,401
	MetaCarvel	5,137	99,831,110	91.29%	68	1,991,282	1,810,249	199,409	158,308

Table 6.2: Comparison of reference based assembly statistics for synthetic metagenomic dataset. We used MEGAHIT and metaSPAdes assembly as the input for OPERA-LG and MetaCarvel. metaSPAdes(S) denotes the scaffolds generated by metaSPAdes assembler.

6.3.5 Evaluation of scaffold quality using synthetic datasets

We evaluated MetaCarvel on a synthetic dataset generated by Singer et al. [208]. This dataset consists of a mixture of 23 bacterial and three archaeal stains, across 10 different phyla and 14 classes, as well as a wide range of GC and repeat content. Due to the high depth of sequencing coverage and relatively low complexity of the mixture, the assembly of the full dataset resulted in large contigs and few opportunities for scaffolding algorithms to improve contiguity. Only 0.051% of mate-pairs spanned the gap between contigs, thereby not providing linking information for scaffolding. To provide a more challenging situation, we downsampled the total number of reads 1000-fold. We assembled the downsampled data using MEGAHIT with default parameters. To derive

linkages between contigs based on mate-pair information, we aligned the reads to the assembled contigs using bowtie2 (with parameters -end-to-end -p 12) [77]. We then used MetaCarvel and OPERA-LG to scaffold these assemblies. Since we know the reference genome sequences for this dataset, we could use metaQUAST [209] to assess the accuracy of the resulting scaffolds. As seen in Table 6.2, MetaCarvel had fewer misassemblies and better contiguity than OPERA-LG, even in this relatively simple community.

We also assembled the data using metaSPAdes (with default parameters), an assembler specifically developed for metagenomic data that also includes a scaffolding module. We scaffolded metaSPAdes contigs with MetaCarvel and OPERA-LG and used metaQUAST to evaluate scaffold accuracy. As seen in Table 6.2, the number of misassemblies in MetaCarvel scaffolds was less than OPERA-LG but more than metaSPAdes scaffolds. MetaSPAdes scaffolds had fewer misassemblies because their scaffolding module is tightly coupled with the assembly module, hence uses more information obtained from the assembly graph to generate scaffolds compared to both OPERA-LG or MetaCarvel. However, the contiguity of MetaCarvel scaffolds was better than both metaSPAdes and OPERA-LG scaffolds.

6.3.6 Evaluation using real metagenomics data

We tested the effectiveness of MetaCarvel on four samples from the Human Microbiome Project (HMP). We chose two stool samples (SRS020233, SRS049959), one supragingival plaque sample (SRR2241598), and a posterior fornix sample (SRS024310). The stool samples represent complex communities and high depths of sequencing cov-

erage, the plaque sample has lower complexity but relatively high coverage, while the posterior fornix has a lower depth of coverage due to the high level of host contamination (more than 80% human DNA) [181]. Table 6.3 shows the comparison of different scaffolding approaches on these samples. Since the composition of these samples is unknown, we could not use reference-based methods to evaluate scaffold accuracy. Instead, as a proxy, we computed the number of mate pairs that map concordantly to the resulting scaffold. We say a mate pair is concordant with the scaffold if the two ends are properly oriented with respect to each other and the distance between the paired reads is within the insert size limit implied by the library. For all the samples, MetaCarvel had a higher number of concordant mate pairs compared to OPERA-LG when the MEGAHIT assembly was used. Even when scaffolding metaSPAdes assemblies, MetaCarvel had the highest number of concordant mate pairs. Also, the total number of concordant mate pairs was higher for both OPERA-LG and MetaCarvel scaffolds when using the MEGAHIT assembly compared to the metaSPAdes assembly as an input. To assess the contiguity scaffolds, we sorted the scaffolds in the decreasing order of their lengths and summed the length of the scaffolds until a particular length was reached (1 Mbp, 10 Mbp, and 50 Mbp in our case). In most cases, MetaCarvel scaffolds had the highest contiguity. Particularly, metaSPAdes contigs scaffolded with MetaCarvel had the highest contiguity. The high contiguity and the high number of concordant mate pairs in MetaCarvel scaffolds can be attributed to its ability resolve the bubbles in the connected components and generate the scaffolds which pass through the bubbles, whereas OPERA-LG broke the scaffolds where there was a boundary between a variant and a linear path (Figure 6.5). Because of this, the concordant mate pairs which aligned at these junctions are not explained by OPERA-LG

scaffolds.

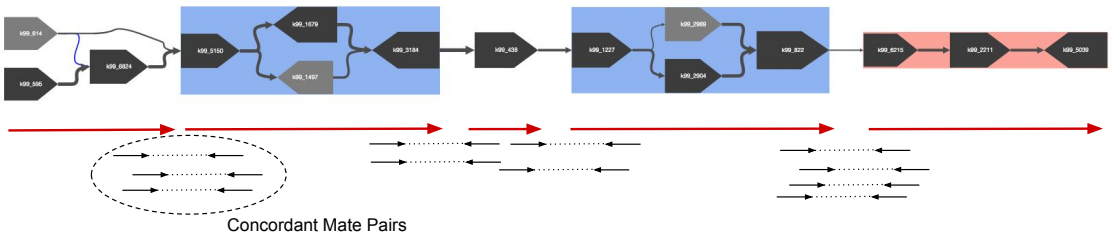


Figure 6.5: A component in the scaffold graph for HMP stool sample. The highlighted nodes in the graph denote the path taken by OPERA-LG to generate the scaffold in this component. It can be observed that while finding a linear path in the graph, other alternative paths are not considered, thereby not using the information provided by the paired-end reads implying these alternative paths.

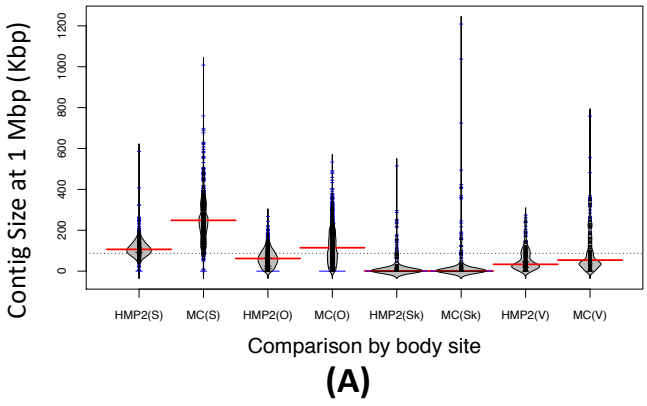
Dataset	Method	Scaffolds	Total Scaffold Size	Scaffolds ≥ 50kbp	Largest Scaffold	Length at 1 Mbp	Length at 10 Mbp	Length at 50 Mbp	Concordant mate pairs
SRS049959	OPERA-LG	198,206	273,240,062	473	530,144	258,645	126,280	38,928	82.15%
	MetaCarvel	108,437	277,078,571	487	518,223	356,683	154,138	39,543	85.49%
	metaSPAdes(S)	98,318	268,262,456	489	476,465	422,765	164,159	44,768	80.05%
	OPERA-LG(M)	97,486	267,725,432	518	476,468	405,096	162,164	47,000	80.08%
	MetaCarvel(M)	98,073	268,106,764	492	868,827	749,803	211,022	47,631	80.32%
SRS020233	OPERA-LG	128,250	279,763,220	393	381,589	286,553	139,736	35,387	84.91%
	MetaCarvel	141,438	282,496,357	421	430,164	356,683	154,277	37,775	86.88%
	metaSPAdes(S)	122,613	279,572,129	437	573,824	351,867	163,997	40,899	85.91%
	OPERA-LG(M)	122,143	279,890,224	459	573,824	372,121	158,298	42,441	85.32%
	MetaCarvel(M)	122,776	280,784,810	471	587,255	584,705	187,147	44,721	85.34%
SRS2241511	OPERA-LG	631	2,844,262	5	96,286	19,320	NA	NA	83.93%
	MetaCarvel	533	2,859,066	6	126,186	27,739	NA	NA	84.24%
	metaSPAdes(S)	774	3,344,834	4	57,030	20,567	NA	NA	91.05%
	OPERA-LG(M)	733	3,343,583	6	124,860	20,978	NA	NA	85.64%
	MetaCarvel(M)	652	3,353,933	11	126,270	37,747	NA	NA	85.87%
SRR2241598	OPERA-LG	60,601	117,644,162	75	217,394	148,289	35,125	41,142	51.59%
	MetaCarvel	56,503	119,165,496	100	319,252	184,177	46,572	5,680	54%
	metaSPAdes(S)	48,403	113,646,647	102	417,398	206,871	46,504	6,223	45.17%
	OPERA-LG(M)	43,908	109,372,182	105	282,943	206,871	47,953	6,613	45.10%
	MetaCarvel(M)	42,927	110,223,597	190	417,398	336,724	97,486	8,703	45.49%

Table 6.3: Comparison of reference free assembly statistics for real metagenomic datasets. (M) denotes the scaffolds generated using metaSPAdes assembly as an input. metaSPAdes(S) denotes the scaffolds generated by metaSPAdes assembler.

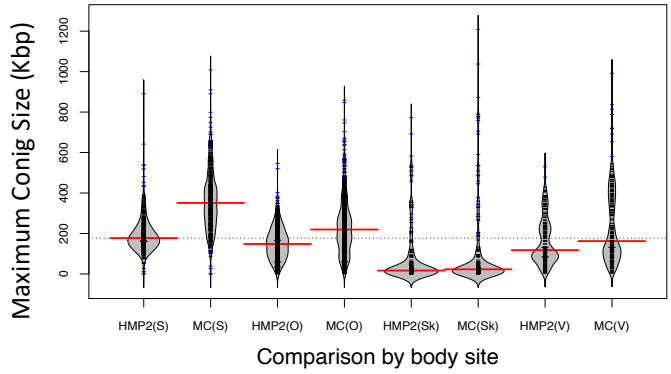
6.4 Scaffolding all samples from Human Microbiome Project (HMP)

To observe the impact of MetaCarvel on the real metagenomic datasets, we ran it on over 2000 samples from the Human Microbiome Project (HMP). We grouped the samples into four main body sites as follows: Oral, Skin, Stool, and Vaginal. We used the assemblies generated using IDBA-UD assembler [66] by HMP consortium. We aligned

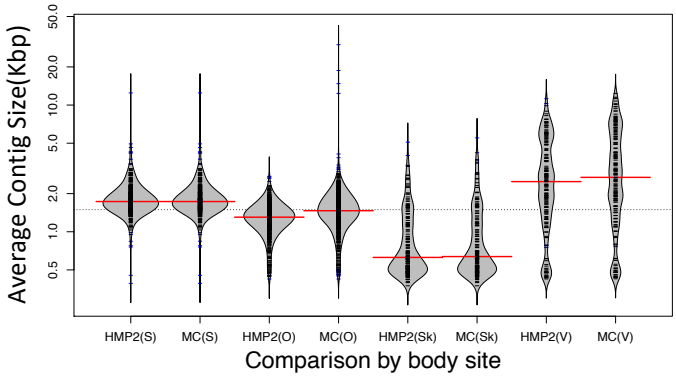
reads to the assemblies with bowtie2 with default parameters and scaffolded them using MetaCarvel. Figure 6.6 shows different statistics computed over all the assemblies. The average contig size for scaffolds generated by MetaCarvel was slightly better than the assembles. However, the maximum contig size was much higher for MetaCarvel, especially for stool, oral and vaginal samples. We also computed the contig size at 1 Mbp length of the assembly. It can be observed that for stool samples, the size at 1 Mbp for MetaCarvel scaffolds is much higher than that for IDBA-UD assemblies. However, for skin samples none of the statistics show significant improvements over the assembly. This can be explained by the coverage of the reads for skin samples. From Table 6.4, it can be observed that the scaffold graph for skin samples is much sparser compared to all other samples (less than one edge per node on an average). This results in the lack of information essential for scaffolding. Since all other samples have more than one edge per node on an average, we saw the improvement in the contiguity after scaffolding. This evaluation highlights the usefulness of scaffolding in getting contiguous scaffolds in the real complex metagenomics samples.



(A)



(B)



(C)

Figure 6.6: Scaffold statistics for HMP samples grouped by different body sites.(A) Contig size at 1 Mbp, (B) Maximum contig size, and (C) Average contig size.

Dataset	Samples	Scaffolds	Average Coverage	Nodes in graph	Edges in graph	Edges per node
HMP2 Stool	427	63813.86	17.09	11820.07	14980.72	1.26
HMP2 Skin	182	13689.73	7.37	390.42	372.46	0.95
HMP2 Oral	1107	56206.90	19.96	19193.04	29265.51	1.52
HMP2 Vaginal	158	20575.69	7.81	974.20	1416.91	1.45

Table 6.4: Graph and sample statistics for HMP2 samples

6.5 Discussion and conclusion

We developed a standalone metagenomics variant detection and scaffolding method MetaCarvel and showed its effectiveness on synthetic and real metagenomics datasets of varying complexity. Unlike most of the existing scaffolders which only output linearized sequences of scaffolds, MetaCarvel outputs a list of variants along with the graph used to call variants. This can help biologists to look at the interesting graph patterns and contig sequences in the assembly and dig down deeper in the biological question related only to the sequences of interest.

We focused our discussion and validation on simple types of genomic variants. Our method is also able to detect more complex variants, which are, however, difficult to validate in an automated fashion. This functionality sets MetaCarvel apart from other tools available for identifying strain variants in microbial communities, tools which primarily rely on reference genomes or conserved genes to characterize microbial strains. The approach taken by MetaCarvel is complementary to approaches based on marker genes, such as StrainPhlAn [195]. The combination of the two approaches represents a promising direction for future research, leading to effective approaches for characterizing novel genomic fragments while placing them within the context of the fine grained taxonomic information derived from marker genes.

The effectiveness of the approach implemented in MetaCarvel critically depends on the data available to the scaffolding module. MetaCarvel can only detect variants if the corresponding contigs are covered at high enough depth and if mate pairs or other information provide links between adjacent contigs. The analysis is also greatly improved if the underlying assembly is conservative - assemblers that aggressively attempt to "smooth out" genomic variants in order to obtain long genomic contigs, end up removing exactly the information that MetaCarvel is designed to detect. We, thus, suggest that scientists interested in strain variation explore multiple assemblies of data sets, using different metagenomic assemblers run with different parameter choices, rather than relying on published assemblies or using the most popular assembler run with default parameters.

In closing, we would like to stress that the study of strain variation within microbial communities is in its infancy, in no small part due to the relative dearth of appropriate data sets and analytic tools. Tools such as MetaCarvel, StrainPhlAn, and others, are just a first step towards the development of an effective toolkit for the discovery and characterization of genomic variants. Of particular interest will be the development of approaches able to infer the functional implications of strain variants, ultimately leading to a better understanding of the principles underlying microbial adaptation and community structure.

Chapter 7: Better greedy sequence clustering with fast banded alignment

7.1 Introduction

The problem of comparing a string against a large set of sequences is of central importance in domains such as computational biology, information retrieval, and databases. Solving this problem is a critical subroutine in many *greedy* clustering heuristics, wherein we iteratively choose a cluster center and form a cluster by recruiting all strings which are similar to the center. In computational biology, sequence similarity search is used to group biological sequences that are closely related. We will use this domain as a motivating example throughout the paper.

Traditionally, clustering 16S rRNA gene [210] sequences involved building a multiple sequence alignment of all sequences, computing a pairwise distance matrix of sequences based on the multiple sequence alignment, and clustering this matrix [211]. However, finding the best multiple sequence alignment is computationally intractable and belongs to the class of NP-hard problems [212]. Another naive way of clustering sequences is to perform an all-versus-all comparison to compute a similarity metric such as edit distance and perform hierarchical clustering to merge closely related sequences. However, the resulting running time is at least quadratic in the total number of sequences. With the development of faster and cheaper DNA sequencing technologies, metagenomic sequenc-

ing datasets can contain over 1 billion short reads [213]. At this scale, both strategies can prove to be very expensive and take months to generate clusters. Heuristic-based methods like greedy clustering are used to counter these computational challenges. While these methods can have worst case quadratic running time, they can run faster in practice [17, 18, 214].

Here, we show a new method for reducing that worst case quadratic running time in practice when the distance metric is the Levenshtein distance [215] and a maximum distance of d determines the similarity. Our algorithm improves the speed of the recruitment step wherein we seek all strings within d distance of a chosen center. In addition to promising experimental results, we give slightly weaker, but provable, guarantees for our techniques while many existing methods do not. Finally, we analyze the quality of the clusters output by our method in comparison to the popular greedy clustering tool UCLUST. We show that the clusters we generate can be both tighter and larger.

7.1.1 Related Work

The problem of comparing a query string against a large string database has been widely studied for at least the past twenty years. For similarity metrics like the edit distance, a dynamic programming algorithm [216] can be used to compare two sequences in $O(m^2)$ time, where m is the length of the sequences. When we only wish to identify strings which are at most edit distance d apart, the running time for each comparison can be reduced to $O(md)$ [217] using a modified version of the standard dynamic programming algorithm. This type of sequence alignment is referred to as *banded alignment* in the

literature since we only need to consider a diagonal "band" through the dynamic programming table. The simple dynamic programming approach can also be sped up by using the Four Russians' method [218, 219], which divides the alignment matrix into small square blocks and uses a lookup table to perform the alignment quickly within each block. This brings the running time down to $O(m^2 \log(\log(m))/\log(m))$ and $O(m^2/\log m)$ for arbitrary and finite alphabets, respectively. Myers [220] considered the similar problem of finding all locations at which a query string of length m matches a substring of a text of length M with at most d differences. They used the bit vector parallelism in hardware to achieve a running time of $O(mM/w)$ where w is the machine word size. However, when used for clustering sequences, these methods need to perform a pairwise comparison of all sequences, thereby incurring the high computational cost of $O(n^2)$ comparisons where n is the total number of sequences.

Sequence search against a database is a basic subroutine in sequence clustering in general and greedy clustering in particular. In greedy approaches, we choose some sequences to be cluster centers and clusters are formed by recruiting other sequences which are similar to the centers. Depending on the approach, we may compare a sequence to recruit against a set of cluster centers or compare a single cluster center against all other sequences, recruiting those within some specified distance. The comparison can be made using any of the methods mentioned above, but in the worst case, existing algorithms may need to perform all pairs banded alignment resulting in $O(n^2md)$ running time on arbitrary input data. However, the interesting property of sequencing data is that most of the sequences generated by the experiments are highly similar to each other. To exploit sequence similarity and reduce the computation performed in dynamic program-

ming, the DNACLUST [18] algorithm lexicographically sorts the sequences and compares sequences against the center sequence in sorted order. Since the adjacent sequences in sorted order share a long prefix, the part of the dynamic programming table corresponding to their longest common prefix remains unchanged, allowing the "free" reuse of that part of the table for further alignments. This method fails when two sequences differ at the start but are otherwise similar. In this case, the entire dynamic programming table needs to be recomputed. The UCLUST [17] algorithm uses the USEARCH [17] algorithm to compare a query sequence against a database of cluster centers. However, the algorithm used by UCLUST makes several heuristic choices in order to speed up the calculation of clusters and thus, the resulting clusters are not guaranteed to satisfy any specific requirement. For example, the distances between sequences assigned to the same cluster should ideally satisfy triangle inequality (ensuring that the cluster diameter is at most twice the radius) and the cluster diameters should be both relatively uniform and within the bounds specified by the user.

7.1.2 Preliminaries

Let the multiset S be the set of n sequences to be clustered. Let m be the maximum length of any sequence in S. For simplicity of exposition and analysis, we will assume throughout most of this paper that all sequences have length exactly m. We also assume m is much smaller than n.

7.1.2.1 Distance metric

We use the same edit distance-based similarity metric as DNACLUST [18], namely

$$\text{similarity} = 1 - \frac{\text{edit distance}}{\text{length of the shorter sequence}}$$

Here, we define edit distance to be Levenshtein distance with uniform penalties for insertions, deletions, and substitutions. The "length of the shorter sequence" refers to the original sequences' lengths without considering gaps inserted by the alignment. We say that two sequences are *similar* if their alignment meets or exceeds a given similarity threshold. Let d be the maximum edit distance between two sequences aligned to the same cluster. This distance is usually computed from a similarity threshold provided by the user, e.g., 97%. Both of our algorithms will be performing *banded alignment* with d as the maximum allowable distance. In this case, if we determine that two sequences have a distance greater than d, we need not report their actual distance.

7.1.2.2 Intervals

Our algorithm involves dividing each sequence into overlapping substrings of length k at regular *intervals*. We formalize the definition of an interval as follows. Given a *period length* p such that $k = p + d + 1$, we divide each sequence into $\lfloor m/p \rfloor$ intervals of length k. For $i \in \{0, 1, \ldots, \lfloor m/p \rfloor - 1\}$, the i^{th} interval starts at index ip inclusive and extends to index $ip + k$ exclusive. We will see that we must choose p to be at least d. However, choosing a larger p may give a better speedup when dealing with highly similar

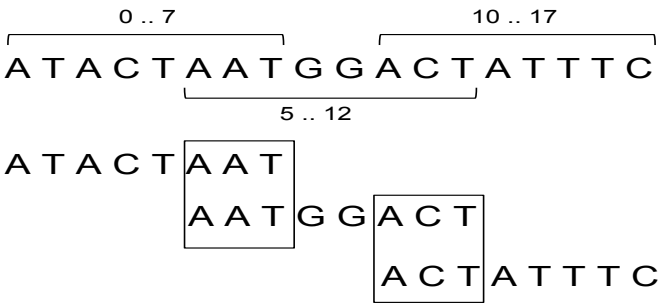


Figure 7.1: An example of how a string is divided in overlapping substrings called intervals. In this case, the length of each substring (k) is 8. Since the substrings must overlap by $d + 1$ characters, which in this case is 3, the period length (p) is 5.

sequences. Further, for an interval i, we define b_i to be the number of *distinct* substrings for interval i over all sequences in $\mathcal{S}$ and we define $b = \max_i b_i$. We will show that when b is much smaller than n we get some theoretical improvement on the running time. Figure 7.1 shows an example of how a sequence is partitioned into a set of overlapping substrings. We store these intervals in a data structure we call an *Edit Distance Interval Trie (EDIT)* which is described in detail in Section 7.2.2.

7.1.2.3 Greedy clustering

The greedy clustering approach (similar to CD-HIT [214], UCLUST, and DNA-CLUST) can be described at a high level as follows. De-replicate the multiset $\mathcal{S}$ to get the set $\mathcal{U}$ of distinct sequences. Optionally, impose some order on $\mathcal{U}$. Then, iteratively remove the top sequence from $\mathcal{U}$ to form a new cluster center s_c. Recruit all sequences $s \in \mathcal{U}$ that are within d distance from s_c. When we recruit a sequence s, we remove it from $\mathcal{U}$ and add it to the cluster centered at s_c. If s_c does not recruit any sequences, we call it a singleton and add it to a list of singletons, rather than clusters. We continue this process until $\mathcal{U}$ is empty.

We order the sequences of $\mathcal{U}$ in decreasing order of their abundance/multiplicity in $\mathcal{S}$. This is also the default ordering used by UCLUST. Alternatively, DNACLUST uses a decreasing order of sequence length. The reason for ordering by abundance is that assuming a random error model, the abundant sequences should be more likely to be "true" centers of a cluster. The reason for DNACLUST ordering by length is to preserve triangle inequality among sequences in a cluster when performing semi-global alignment allowing gaps at the end with no penalty. Semi-global alignment is necessary for comparing reads generated by specific sequencing technologies such as 454. However, since we perform global alignment, triangle inequality is guaranteed regardless of the ordering and thus, ordering by abundance is preferred.

7.1.3 Our Contributions

We developed a method for recruiting in exact greedy clustering inspired by the classical Four Russians' speed up. In Section 7.2, we describe our algorithm and prove that the worst case theoretical running time is better than naive all-versus-all banded alignment under realistic assumptions on the sequencing data used for clustering. In section 7.3, we present experimental results from using our method to cluster a real 16S rRNA gene dataset containing about 2 million distinct sequences. We show that on real data the asymptotic running time of the algorithm grows linearly with the size of the input data. We also evaluated the quality of the clusters generated by our method and compared it with UCLUST, which is one of the widely used methods. We show that our method generates tighter and larger clusters at 99% similarity both when considering edit distance and

evolutionary distance. At 97% similarity, we show that our method produces clusters with a much tighter edit distance diameter compared to UCLUST. While UCLUST runs faster at similarities 97% and less, our approach is faster at higher similarities. In particular, we highlight that UCLUST does not scale linearly at the 99% similarity threshold while our approach does.

7.2 Recruiting algorithm

We show two ways in which the classical Four Russians' speedup can be adapted to banded alignment. Then, we describe a trie-like data structure for storing sequences. Finally, we use this data structure to recruit similar sequences to a given center sequence using our Four Russians' method.

7.2.1 Banded Four Russians' approach

We present two ways to extend the Four Russians' speedup of edit distance computation to banded alignment. The first is a very natural extension of the classical Four Russians' speed up. The second is useful for tailoring our algorithm to meet the needs of 16S rRNA gene clustering. Specifically, we exploit the fact that the strings are similar and the maximum edit distance is small.

7.2.1.1 Warm-up: classic Four Russians' speedup

In the classical Four Russians' speedup of edit distance computation due to to [218, 219], the dynamic programming table is broken up into square *blocks* as shown in the

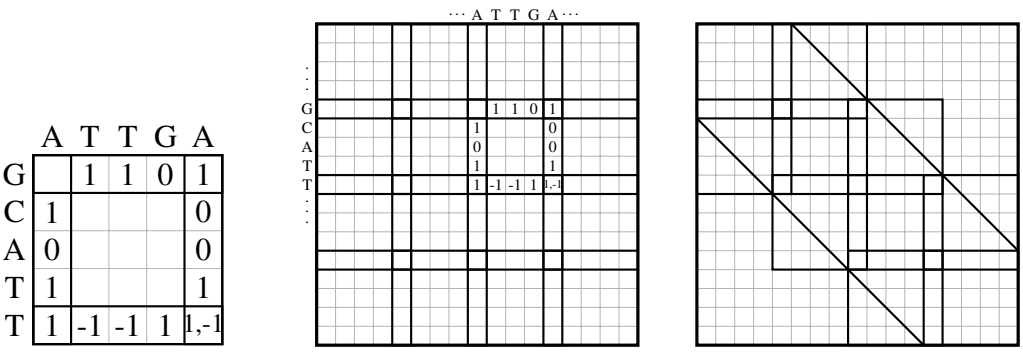


Figure 7.2: Example of classic Four Russians'. **Left:** a single block. Notice that for any input in the upper left corner, we can sum that value with one path along the edges of the block to recover the value in the lower right corner. Note that the offset value in the lower right corner may be different for the row and column vectors overlapping at that cell. In this case, the lower right cell is one more than its left neighbor and one less than its above neighbor. **Center:** the full dynamic programming table divided into nine 5×5 blocks. Note that the offset values in the example block may not correspond to the optimal alignment of the two substrings shown since they depend on the global alignment between the two full-length strings. **Right:** blocks covering only the diagonal band in the context of banded alignment.

center of Fig. 7.2. These blocks are tiled such that they overlap by one column/row on each side (for a thorough description of this technique see [221]). When computing banded alignment, we only need to tile the area within the band as in the right hand of Fig. 7.2. Let the maximum edit distance be d and the string lengths be m. Then our block size k can be as small as $d + 1$, and we require roughly $2m/k$ blocks in total.

The high-level idea of the Four Russians' speedup is to precompute all possible solutions to a *block function* and store them in a lookup table (In our implementation we use lazy computation and store the lookups in a hash table instead of precomputing for all inputs). The block function takes as input the two substrings to be compared in that block and the first row and column of the block itself in the dynamic programming table. It outputs the last row and column of the block. We can see in the Fig. 7.2 that given the two strings and the first row and column of the table, such a function could be applied

repeatedly to compute the lower right cell of the table and therefore, the edit distance. Note that cells outside the band will not be used since any alignment visiting those cells must have distance larger than d.

Several tricks reduce the number of inputs to the block function to bound the time and space requirements of the lookup table. For example, the input row and column for each block can be reduced to vectors in $\{-1, 0, 1\}^d$. These *offset vectors* encode only the difference between one cell and the next (see Fig. 7.2) which is known to be at most 1 in the edit distance table. It has also been shown that the upper left corner does not need to be included in the offset vectors. This bounds the number of possible row and column inputs at 3^d each [218].

Notice that for the banded alignment problem, this may not provide any speedup for comparing just two strings of length m. Indeed, building and querying the lookup table may take more time than simply running the classical dynamic programming algorithm restricted to the band of width $2d + 1$. However, our final algorithm will make many such comparisons between different pairs of strings using the same lookup table. In practice, we also populate the lookup table as needed rather than pre-computing it. This technique, known as *lazy computation*, allows us to avoid adding unnecessary entries for comparisons that don't appear in our dataset. Additionally, decomposing sequences into blocks will be a crucial step in building the data structure in Section 7.2.2.

7.2.1.2 Our approach to the Four Russians' speedup

Notice that the previous approach will not offer many benefits in practice when d is small (e.g., $d = 2$). The overhead of looking up block functions and stitching them together may even be slower than just running dynamic programming on a block. Further, our dataset may not require us to build a lookup table comparing all possible strings of length k.

Here we consider a different block function. This function is designed for situations in which we wish to use a block size k that is larger than $d + 1$. The blocks now overlap on a square of size $d + 1$ at the upper left and lower right corners. We will call these overlapping regions *overlap squares*. Our block function now takes as input the two substrings to be compared and the first row and column of the upper left overlap square. It outputs the first row and column of the lower right overlap square as well as the difference between the upper left corners of the two overlap squares.

Thus, we can move directly from one block to the next, storing a sum of the differences between the upper left corners. In this case, reaching the final lower right cell of the table requires an additional $O(d^2)$ operation to fill in the last overlap square, but this adds only a negligible factor to the running time.

This approach succeeds when some properties of the dataset limit the number of possible substring inputs to the block function as opposed to an absolute theoretical upper bound such as $O(|\sigma|^k)$ based on the number of possible strings of length k for an alphabet σ. Rather than computing and storing all possible inputs, we store the inputs encountered by our algorithm. The advantage is that a larger block size reduces the number of lookups

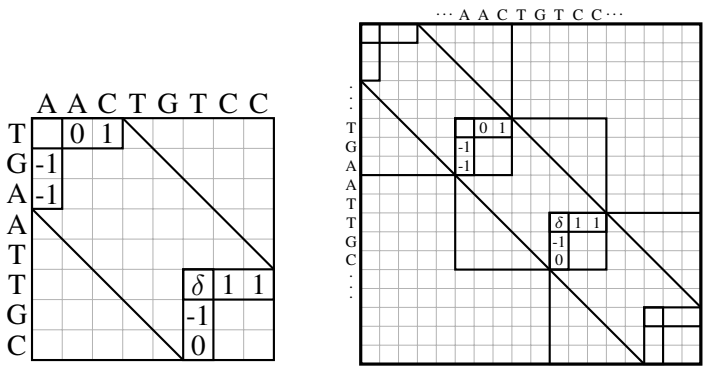


Figure 7.3: Example of our approach to the Four Russians' speed up. **Left:** a block for maximum edit distance $d = 2$. The output δ represents the offset from the upper left corner of the current block to the upper left corner of the next block. Note that we only need to consider a diagonal band of the block itself. **Right:** using these blocks to cover the diagonal band of the dynamic programming table in the context of banded alignment.

needed to compare two strings which is $m/(k - d - 1)$ for this approach. Naturally, the same tricks such as offset encoding of the input rows and columns as some vector in $\{-1, 0, 1\}^d$ can be applied in this case.

Another benefit of this approach is that it is more straightforward to implement in practice. Each block depends on the full output of one previous block. In contrast, the classical approach requires combining partial input from two previous blocks and also sending output to two separate blocks.

7.2.1.3 Theoretical bound on the running time of our approach

To give some intuition, we prove a theoretical bound on the running time under the assumption of at most b distinct substrings per interval in the dataset. This is a reasonable assumption for specific application in computational biology. For example, the 16s rRNA gene is highly conserved, and thus b is much smaller than n for such datasets. While standard banded alignment takes $O(n^2md)$, we show that for small enough b this can be

reduced to $O(n^2m)$. We prove this bound for our approach to using the Four Russians' speedup for banded alignment, but it extends to the classical approach as well.

Theorem 1. *If $b \leq \frac{n}{3^d\sqrt{d}}$, we can find all pairs of distance at most d in $O(n^2m)$ time.*

Proof. To simplify, we will assume the lookup table is pre-computed. Then, we must show that if $b \leq \frac{n}{3^d\sqrt{d}}$, then building the lookup table and doing the actual string comparisons can each be done in $O(n^2m)$ time. We further assume $k \approx 2d$ (in practice we choose a larger k).

First, we show that there are at most $\frac{m}{k-d}b^2 3^{2d}$ entries in the lookup table. There are at most $\frac{m}{k-d}$ intervals and since each interval has at most b distinct strings, there are at most b^2 relevant string comparisons. Each distinct string comparison must be computed for all 3^{2d} offset vector inputs. The cost of generating each lookup entry is simply the cost of computing banded alignment on a block, kd. Thus, the lookup table can be built in time $\frac{m}{k-d}b^2 3^{2d}kd$. Keeping our goal in mind we see that

$$\frac{m}{k-d}b^2 3^{2d}kd \leq n^2m \qquad \text{is true when} \qquad b \leq \frac{n}{3^d\sqrt{d}} \qquad \text{since } k \approx 2d$$

To bound the running time of the string comparisons, notice that comparing two strings requires computing $\frac{m}{k-d}$ block functions. The time spent at each block will be $O(k+d)$ to look up the output of the block function and update our sum for the next corner. Thus, building the lookup table and computing the edit distance between all pairs using the lookup table each takes $O(n^2m)$ time. □

7.2.2 The Edit Distance Interval Trie (EDIT)

To facilitate the crucial step of identifying all strings within edit distance d of a chosen cluster center, we construct a trie-like data structure on the intervals. This structure will be built during a pre-processing stage. Then, during recruitment, any recruited sequences will be deleted from the structure. The procedure for building this structure is summarized in Algorithm 3 and illustrated in Figure 7.4. The main benefit of this data structure, like any trie, is that it exploits prefix similarity to avoid duplicating work.

The mapping in step 2 of Algorithm 3 is a one-to-one mapping to integers from one to the number of distinct substrings. Here, it serves to reduce the size of the data structure since the number of distinct substrings will typically be much less than all possible length k strings on the given alphabet. This mapping also speeds up calls to the lookup table during the recruitment subroutine summarized in the next section.

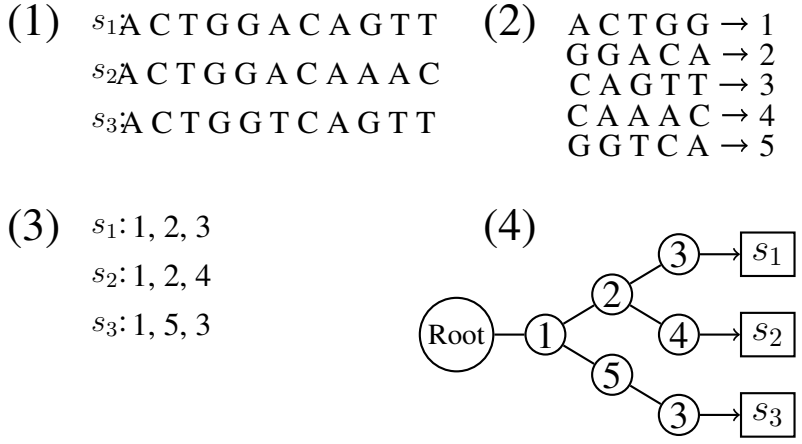
Algorithm 3 BUILD-EDIT

Partition each sequence into overlapping intervals of length k, such that each interval overlaps on exactly $d + 1$ characters.
Map each distinct substring of length k appearing in our list of interval strings to an integer.
Assign an integer vector *signature* to each sequence by replacing each block with its corresponding integer value.
Insert these signatures into a trie with the leaves being pointers to the original sequences.

7.2.3 Recruiting to a center

Given a center sequence s_c, we can recruit all sequences of distance at most d from s_c by simply traversing the trie in depth-first search order and querying the block function of each node we encounter. The input to each block function is the substring of that node

Figure 7.4: Example illustrating the steps of Algorithm 3 with $d = 1$ and $k = 5$.



in the trie, the substring at the same depth within the signature of s_c, and the offset vectors output by the previous block function. As we traverse a path from the root toward a leaf, we store a sum of the edit distance as provided by the output of each block function. If this sum ever exceeds the maximum distance d, we stop exploring that path and backtrack. Whenever we reach a leaf ℓ, we retrieve its corresponding sequence s_ℓ. Then, we align the remaining suffixes and compute the true similarity threshold $d' \leq d$ based on the length of the shorter sequence. If the final edit distance is less than d', we add s_ℓ to the cluster centered at s_c and prune/remove any nodes in the trie corresponding only to s_ℓ.

7.3 Experimental results

7.3.1 Properties of our recruitment algorithm and data structure

In this section, we highlight some key features of our recruitment algorithm and the EDIT data structure. To evaluate our method, we used a dataset consisting of about 57 million 16S rRNA amplicon sequencing reads with 2.7 million distinct sequences. To understand the impact of the number of input sequences to cluster on the average number

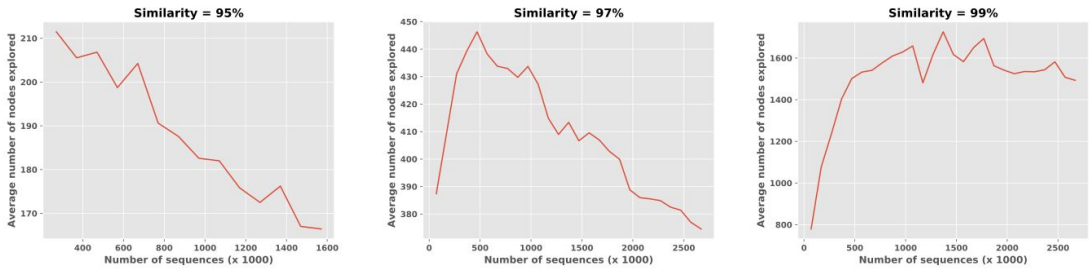


Figure 7.5: Plots for the average number of nodes explored in the tree while recruiting sequences to a cluster center.

of comparisons in each recruitment step, we ran our algorithm on different input sizes at different similarity thresholds. We counted the average number of tree nodes explored while recruiting a particular center sequence and used it as a quantitative representation of the number of comparisons made since all nodes represent a substring of fixed length k. Figure 7.5 shows the plots for the average number of tree nodes explored for different similarity thresholds. For the 95% and 97% similarity thresholds, the average number of nodes explored decreases as more sequences are clustered. This happens because although more sequences are clustered, due to the lower similarity threshold a large number of sequences get clustered in each traversal of the tree. For 99% similarity threshold, the average number of nodes explored increases initially with the number of sequences but becomes uniform after about $100,000$ sequences. The high similarity threshold can explain the strict increase in the number of nodes. However, in all cases, the number of nodes explored by each center does not increase linearly with the number of input sequences. Thus the total number of comparisons made for given dataset is observed to be increasing as a function of n rather than the worst case n^2.

To understand the likelihood of backtracking at each level of the tree, we clustered

a sample of 1.07 million distinct sequences at three different similarity thresholds (95%, 97%, and 99%). The backtracking probability for a given node was calculated as the ratio of the number of times we stopped exploring a path at that node to the total number of times that node was explored. We aggregated this probability for all of the nodes belonging to the same level of the tree. Figure 7.6 shows this likelihood for all levels of the tree at the different similarity thresholds. As we define block size based on the similarity, there are a different number of levels in the tree corresponding to different similarity thresholds. For all three similarity thresholds, the probability of backtracking decreases as we go deeper into the tree except a couple of sharp peaks at intermediate levels. These peaks can be attributed to the sequencing artifacts. The 16S rRNA gene reads are sequenced using the Illumina paired-end sequencing protocol. These paired reads are then merged to make a single read which we use for clustering. The reads contain sequencing errors concentrated near their ends. Due to such sudden errors along the sequence, while recruiting, the edit distance can easily go above the threshold, and backtracking needs to be performed. Since we are using lazy computation, our first encounter with a particular input to the block function requires us to explicitly perform the dynamic programming for that block and store it in the lookup table. However, we observed that this explicit dynamic programming computation is rare and most block functions can be computed by simply querying the lookup table (data not shown).

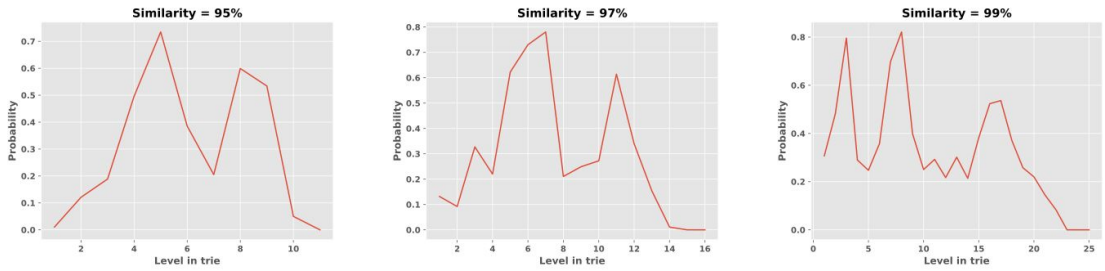


Figure 7.6: Plots for the probability of backtracking at a particular level in the tree. Note that the number of levels is different for different similarity thresholds since our substring length k is dependent on the maximum distance d.

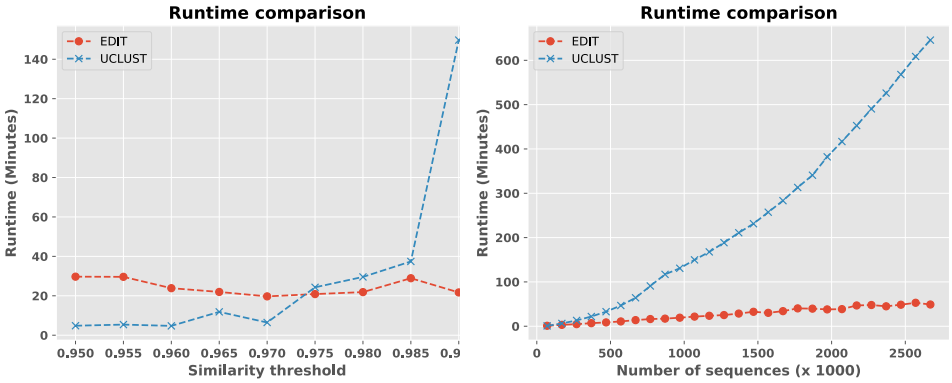


Figure 7.7: Running time comparison of EDIT and UCLUST as a function of similarity threshold and number of sequences.

7.3.2 Comparison with UCLUST

Here, we evaluate EDIT against UCLUST, a highly used tool for analyzing 16S rRNA gene datasets.

7.3.2.1 Running time analysis

We compared the running time of EDIT and UCLUST on a subsample 1.07 million distinct sequences at different similarity thresholds. Figure 7.7 shows the plot for running

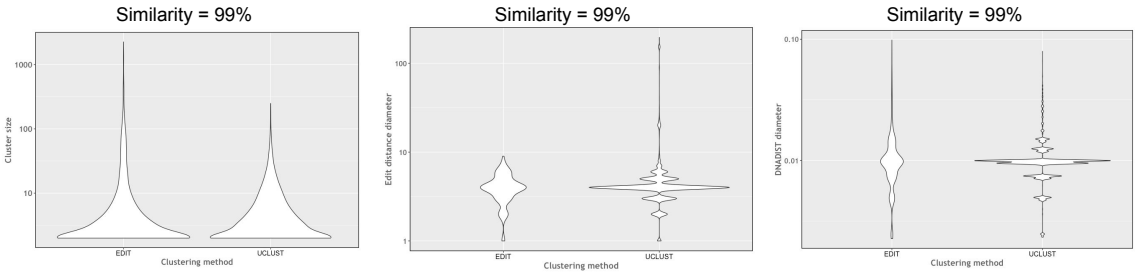


Figure 7.8: Evaluation at similarity threshold of 99%. All the plots are log scaled.

time at different similarity thresholds. We observed that the running time of EDIT stays relatively constant at different similarity thresholds whereas the running time of UCLUST was very low for lower similarity thresholds, but increased non-linearly at higher similarity thresholds. Notably, between 98.5% to 99%, the running time of UCLUST grows five folds. We did further analysis of running time at 99% similarity threshold using different sample sizes as input. Figure 7.7 shows the running time comparison of UCLUST and EDIT. It can be observed that the running time of UCLUST on large sample sizes ($>$ 1 million) grows much faster than the running time of EDIT, which scales almost linearly. For the largest sample of 2.7 million sequences, UCLUST running time was ten times greater than EDIT running time. This evaluation implies that higher similarity thresholds ($>$ 98%), EDIT was faster compared to UCLUST. Also, the running time of EDIT showed low variance compared to UCLUST for different similarity thresholds.

7.3.2.2 Evaluation of clusters

We subsampled 135,880 distinct sequences from the entire dataset and ran both methods at the 97% and 99% similarity thresholds. We then compared the outputs of both methods using three metrics: the cluster size, the cluster diameter based on the sequence

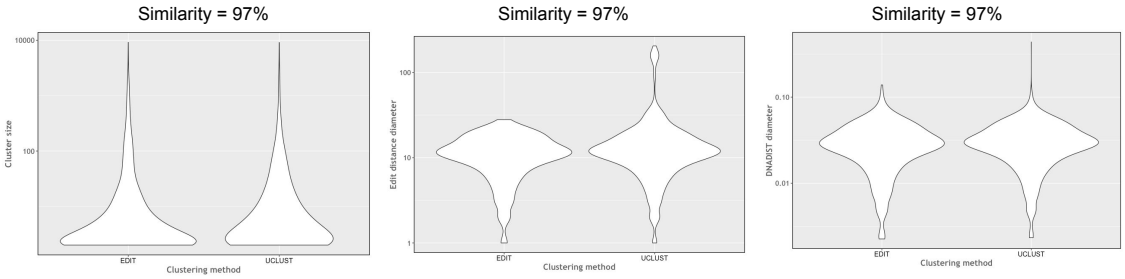


Figure 7.9: Evaluation at similarity threshold of 97%. All the plots are log scaled.

similarity, and the cluster diameter based on the evolutionary distance. To compute the cluster diameter based on sequence similarity, we computed the maximum edit distance between any two sequences in each cluster. To compute the cluster diameter based on evolutionary distance, we first performed a multiple sequence alignment of the sequences in each cluster using clustalW [222]. Once the multiple sequence alignment was computed, we used the DNADIST program from the phylip [223] package to compute a pairwise evolutionary distance matrix. The maximum distance between any pair of sequences is defined as the DNADIST diameter. Using two orthogonal notions of cluster diameter helps to define the "tightness" of clusters. Figures 7.8 and 7.9 show violin plots for different comparison metrics at the 99% and 97% similarity thresholds, respectively.

At the 99% similarity threshold, EDIT can produce larger clusters compared to UCLUST. The edit distance diameters for the clusters generated by EDIT is reasonably well constrained. However, the edit distance diameters for the clusters generated by UCLUST had a significant variance, implying that several different sequences may be getting clustered together. The DNADIST diameter for both methods was comparable. At the 97% similarity threshold, both EDIT and UCLUST generated similar sized clus-

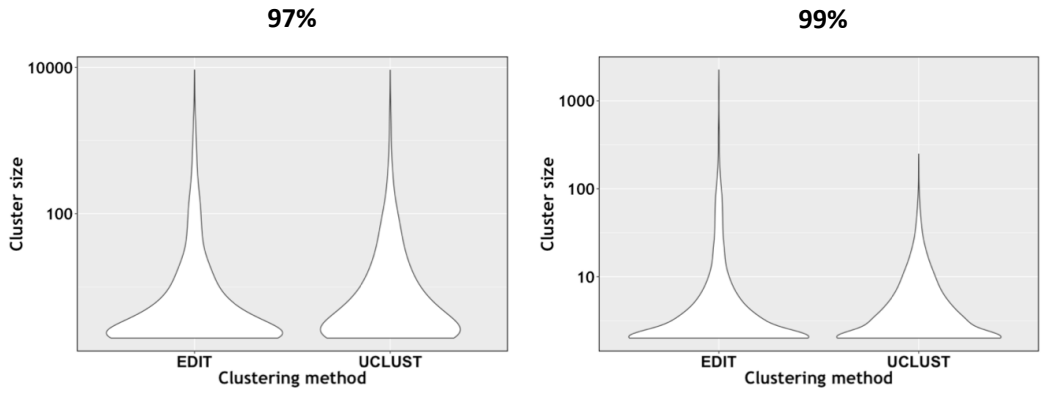


Figure 7.10: Cluster size at 97% and 99% similarity threshold. All the plots are log scaled.

ters. Even in this case, the edit distance diameter for UCLUST clusters showed a larger variance compared to the edit distance diameter for EDIT clusters. The DNADIST diameter for UCLUST has slightly more variance compared to that of EDIT clusters, implying some of the clusters generated by UCLUST had sequences with a considerable evolutionary distance between them. This validation confirms that the sequences in the clusters produced by EDIT at different similarity thresholds are highly similar to each other.

At the 99% similarity threshold, we observed a stark difference between the cluster sizes of EDIT and UCLUST(Figure 7.10). For example, the two largest clusters produced by EDIT had sizes 7,978 and 3,383 respectively whereas the two largest clusters produced by UCLUST were of sizes 249 and 233, which is almost 30 times smaller than the largest EDIT cluster. To investigate this further, we used BLAST [224] to align all clusters of UCLUST against the top two largest clusters of EDIT. We only considered the alignments with 100% alignment identity and alignment coverage. We observed that 765 distinct UCLUST clusters had all of their sequences aligned to the largest EDIT cluster and 837

distinct clusters had at least 80% of their sequences aligned to the largest EDIT cluster. Only 82 UCLUST clusters out of 16,968 total (not including singletons) had less than 80% of their sequences mapped to the largest EDIT cluster. Those 82 clusters accounted for only 255 sequences, roughly 30 times fewer than the number of the sequences in the largest EDIT cluster alone. As far as singletons (the clusters with only one sequence) are concerned, EDIT generated 22,318 singleton clusters whereas UCLUST generated 33,519 singleton clusters. For the size of the sample considered in this analysis, this difference is very significant. This evaluation implies that at a high similarity threshold, heuristic-based methods like UCLUST tend to produce fragmented clusters whereas EDIT was able to capture a higher number of similar sequences in a single cluster.

7.4 Conclusion and future directions

The datasets analyzed by biologists are rapidly increasing in size due to the advancements in sequencing technologies, and efficient clustering is needed to analyze these datasets in reasonable memory and running time. In this paper, we proposed the first step towards this goal by designing a novel data structure to perform banded sequence alignment. We extended the traditional Four Russians' method to perform banded alignment of highly similar sequences and use that to perform greedy clustering of 16S rRNA amplicon sequencing reads. We compared our method to UCLUST and showed that our method generates tight clusters at different similarity thresholds when both string similarity and evolutionary distance are considered. We focused our discussion of results around high similarity clustering ($\geq 97\%$) because no fixed threshold can create biologically mean-

ingful clusters. Our method can generate mathematically well-defined and tight clusters, which can serve as representative clusters from the original data and thus can be used to perform the downstream computationally intensive analysis.

Although we use clustering as a motivating example throughout the paper, our algorithm could be used in a variety of different contexts where highly similar sequences need to be identified from the data. We plan to extend our algorithm to make it parallelized by performing the traversal of each tree branch in parallel. Sequences which end up becoming singletons explore most the tree and this dominated the running time. We plan to explore different methods such as k-mer filters and locality sensitive hashing to flag singletons and exclude them from the recruiting process.

Chapter 8: Other contributions

8.1 A critical analysis of the Integrated Gene Catalog

A reliable catalog of reference genes can significantly facilitate metagenomic analyses of the human gut microbiota. The Integrated Gene Catalog (IGC), described by Li et al. [37], sought to provide an annotated database representing the complete gene content of the human gut microbiome. Designed to be non-redundant and near-comprehensive, the gene catalog contains 9,879,896 gene clusters primarily derived from human stool microbial communities. The authors calculated that the IGC includes close-to-complete sets of genes for most gut microbes and near the saturated coverage of core gene content and functions for the human gut microbiome. The IGC is intended for quantitative characterization of gut microbiota, genes, and gene functions via read mapping and abundance profiling for metagenomic, metatranscriptomic, and metaproteomic human gut microbiome datasets.

In this project, we critically evaluate these claims, highlighting important issues concerning the methodology used to develop the IGC and the gene catalog itself, and how these issues may affect downstream analyses. To do so, we analyzed the data made freely available by the authors of the IGC [37]. This critique is especially relevant considering that the IGC is highly cited and many high-profile studies have used the IGC directly or

the MOCAT pipeline [225], which was used to create the IGC. This includes work as diverse as the Human Microbiome Project [226], research on the structure and function of the ocean microbiome [227], investigating treatments for epithelial tumors [228], work on the correlations between the gut microbiome and diabetes [229, 230] or overall health [231], as well as looking for signs of fecal contamination in water samples from a river [232].

8.2 A succinct four Russians' speedup for edit distance computation and one-against-many banded alignment

The classical Four Russians' speedup for computing edit distance (a.k.a. Levenshtein distance), due to Masek and Paterson [218], involves partitioning the dynamic programming table into k-by-k square blocks and generating a lookup table in $O(\psi^{2k}k^2|\Sigma|^{2k})$ time and $O(\psi^{2k}k|\Sigma|^{2k})$ space for block size k, where ψ depends on the cost function (for unit costs $\psi = 3$) and $|\Sigma|$ is the size of the alphabet. We show that the $O(\psi^{2k}k^2)$ and $O(\psi^{2k}k)$ factors can be improved to $O(k^2 \lg k)$ time and $O(k^2)$ space. Thus, we improve the time and space complexity of that aspect compared to Masek and Paterson [218] and remove the dependence on ψ.

We further show that for certain problems the $O(|\Sigma|^{2k})$ factor can also be reduced. Using this technique, we show a new algorithm for the fundamental problem of one-against-many banded alignment. In particular, comparing one string of length m to n other strings of length m with maximum distance d can be performed in $O(nm + md^2 \lg d + nd^3)$ time. When d is reasonably small, this approaches or meets the current

best theoretic result of $O(nm + nd^2)$ achieved by using the best known pairwise algorithm running in $O(m+d^2)$ time [233,234] while potentially being more practical. It also improves on the standard practical approach which requires $O(nmd)$ time to iteratively run a $O(md)$ time pairwise banded alignment algorithm.

Regarding pairwise comparison, we extend the classic result of Masek and Paterson [218] which computes the edit distance between two strings in $O(m^2/\log m)$ time to remove the dependence on ψ even when edits have arbitrary costs from a penalty matrix. Crochemore, Landau, and Ziv-Ukelson [235] achieved a similar result, also allowing for unrestricted scoring matrices, but with variable-sized blocks. In practical applications of the Four Russians' speedup wherein space efficiency is essential and smaller block sizes k is used (notably $k < |\Sigma|$), Kim, Na, Park, and Sim [236] showed how to remove the dependence on the alphabet size for the unit cost version, generating a lookup table in $O(3^{2k}(2k)!k^2)$ time and $O(3^{2k}(2k)!k)$ space. Combining their work with our result yields an improvement to $O((2k)!k^2 \lg k)$ time and $O((2k)!k^2)$ space.

8.3 Hierarchically visualizing metagenome assembly graphs with metagenomeScope

Manual inspection of sequence assembly graphs can be useful not only as a debugging tool when developing assembly software but also as a way to uncover interesting biological patterns, such as structural differences between the two or more haplotypes being analyzed in a genomic or metagenomic experiment. Current tools for visualizing these graphs [237–239], however, emphasize a high-level representation, based on force-

directed layouts [240], aimed at revealing the broad level quality of an assembly rather than its small-scale structure. As a result, it is difficult for users to piece together a unified understanding of both the high-level structure of the assembly graph and the detailed patterns found within these graphs. We present a new strategy for displaying genome assembly graphs that emphasizes the expected linear structure of genome assemblies and allows a multi-level exploration of the structure of the graph [241]. This approach is implemented in MetagenomeScope, an interactive web-based tool. We detail the novel layout algorithm employed by MetagenomeScope and provide a qualitative comparison with the main tools currently used to explore genome assembly graphs. We demonstrate that MetagenomeScope provides a set of unique capabilities that enable the effective visual exploration of assembly graphs in the context of several common workflows, such as genome finishing or the discovery of structural variants within assemblies.

8.4 A chromosome-scale assembly of *Anopheles funestus*

Many insect genomes remain a challenge to assemble, and mosquito genomes have proven particularly difficult due to their repeat content and structurally dynamic genomes. These issues are compounded by the requirements of emerging, long-read sequencing technologies that typically require >10 μg of DNA for library construction. As a result, it is often impossible to construct a sequencing library from a single individual. Instead, sequencing a pool of individuals from an inbred population is required [242]. For species that are amenable to extensive inbreeding, this approach has led to reference-grade genomes directly from the assembler [243]. However, when inbreeding is not pos-

sible, the sequenced pool of individuals can carry population variation that can fragment the resulting assembly. Instead of assembling a single genome, the assembler must reconstruct some unknown number of varying genomes. In this work, we generated the chromosome-scale assembly of *Anopheles funestus* mosquito using long read sequencing and Hi-C data. Our assembly is 140 times more contiguous than the existing reference genome for *Anopheles funestus*.

8.5 Framework to model human behavior at large ccale during natural disasters

Natural disasters such as hurricanes, floods or tornadoes affect millions of individuals every year. As a result, governments spend millions of dollars in emergency response allocating resources to mitigate the damages. Effective resource allocation requires a deep understanding of how humans react when a disaster takes place. However, gathering human behaviors at a large scale during a disaster is not trivial. For example, some natural disasters like floods might generate temporary displacements or permanent relocations. Drawing a complete picture of such population mobility patterns is extremely difficult. Generally, emergency responders in the field gather data by interviewing affected individuals, but the coverage of these interviews can be pretty limited.

The widespread use of cell phones worldwide has allowed modeling human behaviors at a large scale through the use of Call Detail Records (CDR) [244]. CDRs are collected by cell phone companies for billing purposes every time a phone call is made or received. Each CDR contains information regarding the phone numbers involved in

the communication, date, time and the location (as a pair of latitude and longitude) of the cellular towers that gave coverage to the service. As previous research has shown, CDRs can offer a detailed picture of how humans move and interact with each other [245–248]. In this paper, we propose a novel framework to automatically extract large-scale models of human behavior during disasters using CDRs. The primary objective is to allow emergency responders to understand how humans react to a disaster. The resulting behavioral models will provide valuable information not only to allocate resources once a disaster happens critically but also to enhance emergency planning and prevention.

The proposed framework uses a combination of data mining, n-th order Markov Chain models and statistical analyses to infer normal mobility patterns and social network behaviors from CDR data and to automatically quantify behavioral changes regarding displacements and communication patterns when a disaster happens. Unlike previous work [249], our approach uses CDR data which is sparser (both temporally and spatially) than GPS data and thus more challenging regarding accurate mobility inference. More importantly, it offers the advantage that the framework will be useful in emerging regions with minimal resources where GPS cell phones, let alone GPS collection systems, are a rarity; and where the high penetration rates of cell phones offer the opportunity of modeling mobility at large scale.

Chapter 9: Conclusion

Genome assembly is a critical step in most bioinformatics analysis. Despite tremendous advances in DNA sequencing technologies, the fully automated reconstruction of complete genomes from sequencing data alone remains a distant dream. Not only is it currently challenging to reconstruct entire chromosome arms, but even grouping together genomic contigs belonging to a single chromosome is a significant challenge. The most significant advance towards this goal has been the development of technologies that can capture information about chromosome conformation. Coupled with new long-read sequencing data, chromosome conformation data has resulted in dramatic increases in the size of the genomic segments that can be reconstructed, in some cases comprising entire chromosome arms. It has been suggested that *de novo* assembly might be better than read mapping approaches for uncovering large structural variants, even in the cases where a reference genome is available [250]. This is extremely important for understanding the genetic variations in cancer genomes and other diseases such as autism that frequently contain events such as gene fusion, copy number aberrations, and other large structural variants [251, 252]. Hence, we believe that higher quality genome assemblies can positively impact a wide range of disciplines.

Large-scale genome projects such as the Vertebrate Genome Project (VGP) [161]

aim to generate error-free-reference-quality genome assemblies for 66,000 genomes with minimal manual efforts. Our method SALSA2 has been used for scaffolding genomes with Hi-C data in phase 1 of the project and has provided promising results. The immediate next step in improving SALSA2 is to use different scaffolding information simultaneously along with the assembly graph to generate accurate scaffolds. Since data from different sequencing technologies are generated for each genome in VGP and other genome sequencing projects in general, having such an integrated scaffolding method is highly desired. One more extension to SALSA2 is adding support for haplotype-aware scaffolding to generate complete haplotype assembly of each copy of the chromosome. In the realm of metagenomics, researchers have been analyzing strains in a sample by manual inspection of the contig assembly graph [253–255]. With the methods proposed in this dissertation, such an analysis can be automated and applied to even larger and complex communities. Recently, the human metagenome or the "other human genome" has been sequenced in projects such as the Human Microbiome Project (HMP) [181] with the goal of furthering the understanding of how the microbiome impacts human health and disease. With the amount of both 16s rRNA and whole genome sequencing data generated in these projects, we believe our tools EDIT and MetaCarvel can provide useful insights into these data. Since not all the microbes in these metagenomic samples are well characterized, MetaCarvel's ability to find variants *de novo* in metagenomic samples can shed light on the nature of variation across different samples and body sites.

It is conceivable that in the very near future further developments in genomic technologies will make the automatic reconstruction of mammalian genomes possible. Rather than the end of a road, such developments will enable scientists to tackle even harder

challenges. One such problem is the complete reconstruction of individual haplotypes, particularly in the context of heterogeneous mixtures such as tumors or microbial mixtures and polyploid genomes. The solution to such complex problems will require further developments in both genomic technologies and algorithms. For biologists interested in understanding and analyzing novel genomes and microbial mixtures, our contributions comprise a valuable toolset.

Bibliography

[1] J Craig Venter, Mark D Adams, Eugene W Myers, Peter W Li, Richard J Mural, Granger G Sutton, Hamilton O Smith, Mark Yandell, Cheryl A Evans, Robert A Holt, et al. The sequence of the human genome. *science*, 291(5507):1304–1351, 2001.

[2] International Human Genome Sequencing Consortium et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860, 2001.

[3] Rodger Staden. A strategy of dna sequencing employing computer programs. *Nucleic acids research*, 6(7):2601–2610, 1979.

[4] Jens G Reich, Heinz Drabsch, and Astrid Däumler. On the statistical assessment of similarities in dna sequences. *Nucleic acids research*, 12(13):5529–5543, 1984.

[5] Todd J Treangen and Steven L Salzberg. Repetitive dna and next-generation sequencing: computational challenges and solutions. *Nature Reviews Genetics*, 13(1):36, 2012.

[6] John D Kececioglu and Eugene W Myers. Combinatorial algorithms for dna sequence assembly. *Algorithmica*, 13(1-2):7, 1995.

[7] Granger G Sutton, Owen White, Mark D Adams, and Anthony R Kerlavage. Tigr assembler: A new tool for assembling large shotgun sequencing projects. *Genome Science and Technology*, 1(1):9–19, 1995.

[8] Melissa de la Bastide and W Richard McCombie. Assembling genomic dna sequences with phrap. *Current Protocols in Bioinformatics*, 17(1):11–4, 2007.

[9] William R Jeck, Josephine A Reinhardt, David A Baltrus, Matthew T Hickenbotham, Vincent Magrini, Elaine R Mardis, Jeffery L Dangl, and Corbin D Jones. Extending assembly of short dna sequences to handle error. *Bioinformatics*, 23(21):2942–2944, 2007.

[10] Serafim Batzoglou, David B Jaffe, Ken Stanley, Jonathan Butler, Sante Gnerre, Evan Mauceli, Bonnie Berger, Jill P Mesirov, and Eric S Lander. Arachne: a whole-genome shotgun assembler. *Genome research*, 12(1):177–189, 2002.

[11] Daniel R Zerbino and Ewan Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829, 2008.

[12] Ruibang Luo, Binghang Liu, Yinlong Xie, Zhenyu Li, Weihua Huang, Jianying Yuan, Guangzhu He, Yanxiang Chen, Qi Pan, Yunjie Liu, et al. Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler. *Gigascience*, 1(1):18, 2012.

[13] Jonathan Butler, Iain MacCallum, Michael Kleber, Ilya A Shlyakhter, Matthew K Belmonte, Eric S Lander, Chad Nusbaum, and David B Jaffe. Allpaths: de novo assembly of whole-genome shotgun microreads. *Genome research*, 18(5):810–820, 2008.

[14] Anton Bankevich, Sergey Nurk, Dmitry Antipov, Alexey A Gurevich, Mikhail Dvorkin, Alexander S Kulikov, Valery M Lesin, Sergey I Nikolenko, Son Pham, Andrey D Prjibelski, et al. Spades: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of computational biology*, 19(5):455–477, 2012.

[15] Robert D Fleischmann, Mark D Adams, Owen White, Rebecca A Clayton, Ewen F Kirkness, Anthony R Kerlavage, Carol J Bult, Jean-Francois Tomb, Brian A Dougherty, Joseph M Merrick, et al. Whole-genome random sequencing and assembly of haemophilus influenzae rd. *Science*, 269(5223):496–512, 1995.

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

[17] Robert C Edgar. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, 26(19):2460–2461, 2010.

[18] Mohammadreza Ghodsi, Bo Liu, and Mihai Pop. DNACLUST: accurate and efficient clustering of phylogenetic marker genes. *BMC bioinformatics*, 12(1):271, 2011.

[19] Qiong Wang, George M Garrity, James M Tiedje, and James R Cole. Naive bayesian classifier for rapid assignment of rrna sequences into the new bacterial taxonomy. *Applied and environmental microbiology*, 73(16):5261–5267, 2007.

[20] J Gregory Caporaso, Justin Kuczynski, Jesse Stombaugh, Kyle Bittinger, Frederic D Bushman, Elizabeth K Costello, Noah Fierer, Antonio Gonzalez Pena, Julia K Goodrich, Jeffrey I Gordon, et al. Qiime allows analysis of high-throughput community sequencing data. *Nature methods*, 7(5):335, 2010.

[21] Patrick D Schloss, Sarah L Westcott, Thomas Ryabin, Justine R Hall, Martin Hartmann, Emily B Hollister, Ryan A Lesniewski, Brian B Oakley, Donovan H Parks, Courtney J Robinson, et al. Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Applied and environmental microbiology*, 75(23):7537–7541, 2009.

[22] Jesse R Zaneveld, Catherine Lozupone, Jeffrey I Gordon, and Rob Knight. Ribosomal rna diversity predicts genome diversity in gut bacteria and their relatives. *Nucleic acids research*, 38(12):3869–3879, 2010.

[23] Shujiro Okuda, Yuki Tsuchiya, Chiho Kiriyama, Masumi Itoh, and Hisao Morisaki. Virtual metagenome reconstruction from 16s rrna gene sequences. *Nature communications*, 3:1203, 2012.

[24] Luke R Thompson, Jon G Sanders, Daniel McDonald, Amnon Amir, Joshua Ladau, Kenneth J Locey, Robert J Prill, Anupriya Tripathi, Sean M Gibbons, Gail Ackermann, et al. A communal catalogue reveals earth?s multiscale microbial diversity. *Nature*, 551(7681), 2017.

[25] Régis Bonnet, Antonia Suau, Joël Doré, Glenn R Gibson, and Matthew D Collins. Differences in rdna libraries of faecal bacteria derived from 10-and 25-cycle pcrs. *International journal of systematic and evolutionary microbiology*, 52(3):757–763, 2002.

[26] Morgan GI Langille, Jesse Zaneveld, J Gregory Caporaso, Daniel McDonald, Dan Knights, Joshua A Reyes, Jose C Clemente, Deron E Burkepile, Rebecca L Vega Thurber, Rob Knight, et al. Predictive functional profiling of microbial communities using 16s rrna marker gene sequences. *Nature biotechnology*, 31(9):814, 2013.

[27] Kathrin P Aßhauer, Bernd Wemheuer, Rolf Daniel, and Peter Meinicke. Tax4fun: predicting functional profiles from metagenomic 16s rrna data. *Bioinformatics*, 31(17):2882–2884, 2015.

[28] Sahar Abubucker, Nicola Segata, Johannes Goll, Alyxandria M Schubert, Jacques Izard, Brandi L Cantarel, Beltran Rodriguez-Mueller, Jeremy Zucker, Mathangi Thiagarajan, Bernard Henrissat, et al. Metabolic reconstruction for metagenomic data and its application to the human microbiome. *PLoS computational biology*, 8(6):e1002358, 2012.

[29] Matthias Scholz, Doyle V Ward, Edoardo Pasolli, Thomas Tolio, Moreno Zolfo, Francesco Asnicar, Duy Tin Truong, Adrian Tett, Ardythe L Morrow, and Nicola Segata. Strain-level microbial epidemiology and population genomics from shotgun metagenomics. *Nature methods*, 13(5):435, 2016.

[30] Johannes Alneberg, Brynjar Smári Bjarnason, Ino de Bruijn, Melanie Schirmer, Joshua Quick, Umer Z Ijaz, Nicholas J Loman, Anders F Andersson, and Christopher Quince. Concoct: clustering contigs on coverage and composition. *arXiv preprint arXiv:1312.4038*, 2013.

[31] Johannes Alneberg, Brynjar Smári Bjarnason, Ino De Bruijn, Melanie Schirmer, Joshua Quick, Umer Z Ijaz, Leo Lahti, Nicholas J Loman, Anders F Andersson, and Christopher Quince. Binning metagenomic contigs by coverage and composition. *Nature methods*, 11(11):1144, 2014.

[32] Steven L Salzberg, Arthur L Delcher, Simon Kasif, and Owen White. Microbial gene identification using interpolated markov models. *Nucleic acids research*, 26(2):544–548, 1998.

[33] Alexander V Lukashin and Mark Borodovsky. Genemark. hmm: new solutions for gene finding. *Nucleic acids research*, 26(4):1107–1115, 1998.

[34] Stefan Götz, Juan Miguel García-Gómez, Javier Terol, Tim D Williams, Shivashankar H Nagaraj, María José Nueda, Montserrat Robles, Manuel Talón, Joaquín Dopazo, and Ana Conesa. High-throughput functional annotation and data mining with the blast2go suite. *Nucleic acids research*, 36(10):3420–3435, 2008.

[35] Yaniv Loewenstein, Domenico Raimondo, Oliver C Redfern, James Watson, Dmitrij Frishman, Michal Linial, Christine Orengo, Janet Thornton, and Anna Tramontano. Protein function annotation by homology-based inference. *Genome biology*, 10(2):207, 2009.

[36] Junjie Qin, Ruiqiang Li, Jeroen Raes, Manimozhiyan Arumugam, Kristoffer Solvsten Burgdorf, Chaysavanh Manichanh, Trine Nielsen, Nicolas Pons, Florence Levenez, Takuji Yamada, et al. A human gut microbial gene catalogue established by metagenomic sequencing. *nature*, 464(7285):59, 2010.

[37] Junhua Li, Huijue Jia, Xianghang Cai, Huanzi Zhong, Qiang Feng, Shinichi Sunagawa, Manimozhiyan Arumugam, Jens Roat Kultima, Edi Prifti, Trine Nielsen, et al. An integrated catalog of reference genes in the human gut microbiome. *Nature biotechnology*, 32(8):834, 2014.

[38] Liang Xiao, Qiang Feng, Suisha Liang, Si Brask Sonne, Zhongkui Xia, Xinmin Qiu, Xiaoping Li, Hua Long, Jianfeng Zhang, Dongya Zhang, et al. A catalog of the mouse gut metagenome. *Nature biotechnology*, 33(10):1103, 2015.

[39] Luis Pedro Coelho, Jens Roat Kultima, Paul Igor Costea, Coralie Fournier, Yuanlong Pan, Gail Czarnecki-Maulden, Matthew Robert Hayward, Sofia K Forslund, Thomas Sebastian Benedikt Schmidt, Patrick Descombes, et al. Similarity of the dog and human gut microbiomes in gene content and response to diet. *Microbiome*, 6(1):72, 2018.

[40] BD Williams, B Schrank, C Huynh, R Shownkeen, and RH Waterston. A genetic mapping system in caenorhabditis elegans based on polymorphic sequence-tagged sites. *Genetics*, 131(3):609–624, 1992.

[41] Rina Wu and ZR Shi. Comparison of chromogenic in situ hybridization, fluorescence in situ hybridization, and immunohistochemistry. *Handbook of IHC and ISH of Human Carcinomas: Molecular Genetics*, pages 13–21, 2004.

[42] S Lawrence, NE Morton, and DR Cox. Radiation hybrid mapping. *Proceedings of the National Academy of Sciences*, 88(17):7477–7480, 1991.

[43] David C Schwartz, Xiaojun Li, Luis I Hernandez, Satyadarshan P Ramnarain, Edward J Huff, and Yu-Ker Wang. Ordered restriction maps of saccharomyces cerevisiae chromosomes constructed by optical mapping. *Science*, 262(5130):110–114, 1993.

[44] David C Schwartz. Microbial genome program report: Optical approaches for physical mapping and sequence assembly of the deinococcus radiodurans chromosome. Technical report, New York Univ.(US), 1999.

[45] Ernest T Lam, Alex Hastie, Chin Lin, Dean Ehrlich, Somes K Das, Michael D Austin, Paru Deshpande, Han Cao, Niranjan Nagarajan, Ming Xiao, et al. Genome mapping on nanochannel arrays for structural variation analysis and sequence assembly. *Nature biotechnology*, 30(8):771, 2012.

[46] James W Fickett and Michael J Cinkosky. A genetic algorithm for assembling chromosome physical maps. In *Bioinformatics, Supercomputing and Complex Genome Analysis*, pages 273–285. World Scientific, 1993.

[47] Will Gillett, Jim Daues, Liz Hanks, and Rob Capra. Fragment collapsing and splitting while assembling high-resolution restriction maps. *Journal of Computational Biology*, 2(2):185–205, 1995.

[48] Yuji Kohara, Kiyotaka Akiyama, and Katsumi Isono. The physical map of the whole e. coli chromosome: application of a new strategy for rapid analysis and sorting of a large genomic library. *Cell*, 50(3):495–508, 1987.

[49] Friedrich W Engler, James Hatfield, William Nelson, and Carol A Soderlund. Locating sequence on fpc maps and selecting a minimal tiling path. *Genome Research*, 13(9):2152–2163, 2003.

[50] Martin Charles Golumbic, Haim Kaplan, and Ron Shamir. On the complexity of dna physical mapping. *Advances in Applied Mathematics*, 15(3):251–261, 1994.

[51] Carol Soderlund, Ian Longden, and Richard Mott. Fpc: a system for building contigs from restriction fingerprinted clones. *Bioinformatics*, 13(5):523–535, 1997.

[52] Thomas S Anantharaman, Bud Mishra, and David C Schwartz. Genomics via optical mapping ii: Ordered restriction maps. *Journal of Computational Biology*, 4(2):91–118, 1997.

[53] Anton Valouev, David C Schwartz, Shiguo Zhou, and Michael S Waterman. An algorithm for assembly of ordered restriction maps from single dna molecules. *Proceedings of the National Academy of Sciences*, 103(43):15770–15775, 2006.

[54] Niranjan Nagarajan, Timothy D Read, and Mihai Pop. Scaffolding and validation of bacterial genome assemblies using optical restriction maps. *Bioinformatics*, 24(10):1229–1235, 2008.

[55] Martin D Muggli, Simon J Puglisi, and Christina Boucher. Efficient indexed alignment of contigs to optical maps. In *International Workshop on Algorithms in Bioinformatics*, pages 68–81. Springer, 2014.

[56] Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 390–398. IEEE, 2000.

[57] Lee M Mendelowitz, David C Schwartz, and Mihai Pop. Maligner: a fast ordered restriction map aligner. *Bioinformatics*, 32(7):1016–1022, 2015.

[58] Daniel H Huson, Knut Reinert, and Eugene W Myers. The greedy path-merging algorithm for contig scaffolding. *Journal of the ACM (JACM)*, 49(5):603–615, 2002.

[59] Leena Salmela, Veli Mäkinen, Niko Välimäki, Johannes Ylinen, and Esko Ukkonen. Fast scaffolding with small independent mixed integer programs. *Bioinformatics*, 27(23):3259–3265, 2011.

[60] Adel Dayarian, Todd P Michael, and Anirvan M Sengupta. Sopra: Scaffolding algorithm for paired reads via statistical optimization. *BMC bioinformatics*, 11(1):345, 2010.

[61] Nilgun Donmez and Michael Brudno. Scarpa: scaffolding reads with practical algorithms. *Bioinformatics*, 29(4):428–434, 2012.

[62] Mihai Pop, Daniel S Kosack, and Steven L Salzberg. Hierarchical scaffolding with bambus. *Genome research*, 14(1):149–159, 2004.

[63] Marten Boetzer, Christiaan V Henkel, Hans J Jansen, Derek Butler, and Walter Pirovano. Scaffolding pre-assembled contigs using sspace. *Bioinformatics*, 27(4):578–579, 2010.

[64] Song Gao, Denis Bertrand, Burton KH Chia, and Niranjan Nagarajan. Opera-lg: efficient and exact scaffolding of large, repeat-rich eukaryotic genomes with performance guarantees. *Genome biology*, 17(1):102, 2016.

[65] Jared T Simpson, Kim Wong, Shaun D Jackman, Jacqueline E Schein, Steven JM Jones, and Inanç Birol. Abyss: a parallel assembler for short read sequence data. *Genome research*, 19(6):1117–1123, 2009.

[66] Yu Peng, Henry CM Leung, Siu-Ming Yiu, and Francis YL Chin. Idba-ud: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics*, 28(11):1420–1428, 2012.

[67] Sergey Nurk, Dmitry Meleshko, Anton Korobeynikov, and Pavel A Pevzner. metaspades: a new versatile metagenomic assembler. *Genome research*, 27(5):824–834, 2017.

[68] Andrey D Prjibelski, Irina Vasilinetc, Anton Bankevich, Alexey Gurevich, Tatiana Krivosheeva, Sergey Nurk, Son Pham, Anton Korobeynikov, Alla Lapidus, and Pavel A Pevzner. Exspander: a universal repeat resolver for dna fragment assembly. *Bioinformatics*, 30(12):i293–i301, 2014.

[69] Joshua Wetzel, Carl Kingsford, and Mihai Pop. Assessing the benefits of using mate-pairs to resolve repeats in de novo short-read prokaryotic assemblies. *BMC bioinformatics*, 12(1):95, 2011.

[70] Erez Lieberman-Aiden, Nynke L Van Berkum, Louise Williams, Maxim Imakaev, Tobias Ragoczy, Agnes Telling, Ido Amit, Bryan R Lajoie, Peter J Sabo, Michael O Dorschner, et al. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *science*, 326(5950):289–293, 2009.

[71] Nicholas H Putnam, Brendan L O'Connell, Jonathan C Stites, Brandon J Rice, et al. Chromosome-scale shotgun assembly using an in vitro method for long-range linkage. *Genome research*, 26(3):342–350, 2016.

[72] Steven Wingett, Philip Ewels, Mayra Furlan-Magaril, Takashi Nagano, Stefan Schoenfelder, Peter Fraser, and Simon Andrews. Hicup: pipeline for mapping and processing hi-c data. *F1000Research*, 4, 2015.

[73] Nicolas Servant, Nelle Varoquaux, Bryan R Lajoie, Eric Viara, Chong-Jian Chen, Jean-Philippe Vert, Edith Heard, Job Dekker, and Emmanuel Barillot. Hic-pro: an optimized and flexible pipeline for hi-c data processing. *Genome biology*, 16(1):259, 2015.

[74] Neva C Durand, Muhammad S Shamim, et al. Juicer provides a one-click system for analyzing loop-resolution hi-c experiments. *Cell systems*, 3(1):95–98, 2016.

[75] Neva C Durand, James T Robinson, Muhammad S Shamim, et al. Juicebox provides a visualization system for hi-c contact maps with unlimited zoom. *Cell systems*, 3(1):99–101, 2016.

[76] Michael EG Sauria, Jennifer E Phillips-Cremins, Victor G Corces, and James Taylor. Hifive: a tool suite for easy and efficient hic and 5c data analysis. *Genome biology*, 16(1):237, 2015.

[77] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357, 2012.

[78] Heng Li and Richard Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[79] Matei Zaharia, William J Bolosky, Kristal Curtis, Armando Fox, David Patterson, Scott Shenker, Ion Stoica, Richard M Karp, and Taylor Sittler. Faster and more accurate sequence alignment with snap. *arXiv preprint arXiv:1111.5572*, 2011.

[80] Noam Kaplan and Job Dekker. High-throughput genome scaffolding from in vivo dna interaction frequency. *Nature biotechnology*, 31(12):1143–1147, 2013.

[81] Sante Gnerre, Iain MacCallum, Dariusz Przybylski, Filipe J Ribeiro, Joshua N Burton, Bruce J Walker, Ted Sharpe, Giles Hall, Terrance P Shea, Sean Sykes, et al. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proceedings of the National Academy of Sciences*, 108(4):1513–1518, 2011.

[82] John D Head and Michael C Zerner. A broydenfletchergoldfarbshanno optimization procedure for molecular geometries. *Chemical physics letters*, 122(3):264–270, 1985.

[83] Hervé Marie-Nelly, Martial Marbouty, Axel Cournac, Jean-François Flot, Gianni Liti, Dante Poggi Parodi, Sylvie Syan, Nancy Guillén, Antoine Margeot, Christophe Zimmer, et al. High-quality genome (re) assembly using chromosomal contact data. *Nature communications*, 5:5695, 2014.

[84] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995.

[85] Jay Ghurye, Mihai Pop, Sergey Koren, Derek Bickhart, and Chen-Shan Chin. Scaffolding of long read assemblies using long range contact information. *BMC genomics*, 18(1):527, 2017.

[86] Olga Dudchenko, Sanjit S Batra, Arina D Omer, et al. De novo assembly of the aedes aegypti genome using hi-c yields chromosome-length scaffolds. *Science*, 356(6333):92–95, 2017.

[87] Dirk Holste, Ivo Grosse, and Hanspeter Herzel. Statistical analysis of the dna sequence of human chromosome 22. *Physical Review E*, 64(4):041917, 2001.

[88] Yulia Mostovoy, Michal Levy-Sakin, Jessica Lam, Ernest T Lam, Alex R Hastie, Patrick Marks, Joyce Lee, Catherine Chu, Chin Lin, Željko Džakula, et al. A hybrid approach for de novo human genome sequence assembly and phasing. *Nature methods*, 13(7):587, 2016.

[89] Anton Bankevich and Pavel A Pevzner. Truspades: barcode assembly of truseq synthetic long reads. *Nature methods*, 13(3):248, 2016.

[90] Andrew Adey, Jacob O Kitzman, Joshua N Burton, Riza Daza, Akash Kumar, Lena Christiansen, Mostafa Ronaghi, Sasan Amini, Kevin L Gunderson, Frank J Steemers, et al. In vitro, long-range sequence information for de novo genome assembly via transposase contiguity. *Genome research*, 24(12):2041–2049, 2014.

[91] Neil I Weisenfeld, Vijay Kumar, Preyas Shah, Deanna M Church, and David B Jaffe. Direct determination of diploid genome sequences. *Genome research*, 27(5):757–767, 2017.

[92] Arthur L Delcher, Steven L Salzberg, and Adam M Phillippy. Using mummer to identify similar regions in large sequence sets. *Current Protocols in Bioinformatics*, pages 10–3, 2003.

[93] Amir-Mohammad Rahmani, Pasi Liljeberg, Juha Plosila, and Hannu Tenhunen. Lastz: An ultra optimized 3d networks-on-chip architecture. In *Digital System Design (DSD), 2011 14th Euromicro Conference on*, pages 173–180. IEEE, 2011.

[94] Daniel C Richter, Stephan C Schuster, and Daniel H Huson. Oslay: optimal syntenic layout of unfinished assemblies. *Bioinformatics*, 23(13):1573–1579, 2007.

[95] Samuel Assefa, Thomas M Keane, Thomas D Otto, Chris Newbold, and Matthew Berriman. Abacas: algorithm-based automatic contiguation of assembled sequences. *Bioinformatics*, 25(15):1968–1969, 2009.

[96] Anna I Rissman, Bob Mau, Bryan S Biehl, Aaron E Darling, Jeremy D Glasner, and Nicole T Perna. Reordering contigs of draft genomes using the mauve aligner. *Bioinformatics*, 25(16):2071–2073, 2009.

[97] Chi-Long Li, Kun-Tze Chen, and Chin Lung Lu. Assembling contigs in draft genomes using reversals and block-interchanges. In *BMC bioinformatics*, volume 14, page S9. BioMed Central, 2013.

[98] Peter Husemann and Jens Stoye. r2cat: synteny plots and comparative assembly. *Bioinformatics*, 26(4):570–571, 2009.

[99] Chin Lung Lu, Kun-Tze Chen, Shih-Yuan Huang, and Hsien-Tai Chiu. Car: contig assembly of prokaryotic draft genomes using rearrangements. *BMC bioinformatics*, 15(1):381, 2014.

[100] Emanuele Bosi, Beatrice Donati, Marco Galardini, Sara Brunetti, Marie-France Sagot, Pietro Lió, Pierluigi Crescenzi, Renato Fani, and Marco Fondi. Medusa: a multi-draft based scaffolder. *Bioinformatics*, 31(15):2443–2451, 2015.

[101] Mikhail Kolmogorov, Brian Raney, Benedict Paten, and Son Pham. Ragouta reference-assisted assembly tool for bacterial genomes. *Bioinformatics*, 30(12):i302–i309, 2014.

[102] Feng Zeng, Lan Yao, Zhigang Chen, and Huamei Qi. A distributed and shortest-path-based algorithm for maximum cover sets problem in wireless sensor networks. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 1224–1228. IEEE, 2011.

[103] Max Alekseyev and Pavel A Pevzner. Breakpoint graphs and ancestral genome reconstructions. *Genome research*, pages gr–082784, 2009.

[104] Tandy Warnow. *Computational phylogenetics: An introduction to designing methods for phylogeny estimation*. Cambridge University Press, 2017.

[105] Kun-Tze Chen, Cheih-Jung Chen, Hsin-Ting Shen, Chia-Liang Liu, Shang-Hao Huang, and Chin Lung Lu. Multi-car: a tool of contig scaffolding using multiple references. *BMC bioinformatics*, 17(17):469, 2016.

[106] Michael J Levene, Jonas Korlach, Stephen W Turner, Mathieu Foquet, Harold G Craighead, and Watt W Webb. Zero-mode waveguides for single-molecule analysis at high concentrations. *science*, 299(5607):682–686, 2003.

[107] Miten Jain, Hugh E Olsen, Benedict Paten, and Mark Akeson. The oxford nanopore minion: delivery of nanopore sequencing to the genomics community. *Genome biology*, 17(1):239, 2016.

[108] Marten Boetzer and Walter Pirovano. Sspace-longread: scaffolding bacterial draft genomes using long read sequence information. *BMC bioinformatics*, 15(1):211, 2014.

[109] Mark J Chaisson and Glenn Tesler. Mapping single molecule sequencing reads using basic local alignment with successive refinement (blasr): application and theory. *BMC bioinformatics*, 13(1):238, 2012.

[110] René L Warren, Chen Yang, Benjamin P Vandervalk, Bahar Behsaz, Albert Lagman, Steven JM Jones, and Inanç Birol. Links: Scalable, alignment-free scaffolding of draft genomes with long reads. *GigaScience*, 4(1):35, 2015.

[111] Marten Boetzer and Walter Pirovano. Toward almost closed genomes with gapfiller. *Genome biology*, 13(6):R56, 2012.

[112] Daniel Paulino, René L Warren, Benjamin P Vandervalk, Anthony Raymond, Shaun D Jackman, and Inanç Birol. Sealer: a scalable gap-closing application for finishing draft genomes. *BMC bioinformatics*, 16(1):230, 2015.

[113] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. *Internet mathematics*, 1(4):485–509, 2004.

[114] Pavel A Pevzner, Haixu Tang, and Michael S Waterman. An eulerian path approach to dna fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17):9748–9753, 2001.

[115] Adam C English, Stephen Richards, Yi Han, Min Wang, Vanesa Vee, Jiaxin Qu, Xiang Qin, Donna M Muzny, Jeffrey G Reid, Kim C Worley, et al. Mind the gap: upgrading genomes with pacific biosciences rs long-read sequencing technology. *PloS one*, 7(11):e47768, 2012.

[116] Shunichi Kosugi, Hideki Hirakawa, and Satoshi Tabata. Gmcloser: closing gaps in assemblies accurately with a likelihood-based selection of contig or long-read alignments. *Bioinformatics*, 31(23):3733–3741, 2015.

[117] Rebecca R Murphy, Jared OConnell, Anthony J Cox, and Ole Schulz-Trieglaff. Nxrepair: error correction in de novo sequence assembly using nextera mate pairs. *PeerJ*, 3:e996, 2015.

[118] Derek M Bickhart, Benjamin D Rosen, Sergey Koren, et al. Single-molecule sequencing and chromatin conformation capture enable de novo reference assembly of the domestic goat genome. *Nature Genetics*, 49(4):643–650, 2017.

[119] Lee Mendelowitz and Mihai Pop. Computational methods for optical mapping. *GigaScience*, 3(1):33, 2014.

[120] Matthew Pendleton, Robert Sebra, Andy Wing Chun Pang, et al. Assembly and diploid architecture of an individual human genome via single-molecule technologies. *Nature methods*, 2015.

[121] Jeong-Sun Seo, Arang Rhie, Junsoo Kim, Sangjin Lee, Min-Hwan Sohn, Chang-Uk Kim, Alex Hastie, Han Cao, Ji-Young Yun, Jihye Kim, et al. De novo assembly and phasing of a korean human genome. *Nature*, 538(7624):243, 2016.

[122] Martin Mascher, Heidrun Gundlach, Axel Himmelbach, Sebastian Beier, Sven O Twardziok, Thomas Wicker, Volodymyr Radchuk, Christoph Dockter, Pete E Hedley, Joanne Russell, et al. A chromosome conformation capture ordered sequence of the barley genome. *Nature*, 544(7651):427, 2017.

[123] Huilong Du, Ying Yu, Yanfei Ma, Qiang Gao, Yinghao Cao, Zhuo Chen, Bin Ma, Ming Qi, Yan Li, Xianfeng Zhao, et al. Sequencing and de novo assembly of a near complete indica rice genome. *Nature communications*, 8:15324, 2017.

[124] Wen-Biao Jiao, Gonzalo Garcia Accinelli, Benjamin Hartwig, Christiane Kiefer, David Baker, Edouard Severing, Eva-Maria Willing, Mathieu Piednoel, Stefan Woetzel, Eva Madrid-Herrero, et al. Improving and correcting the contiguity of long-read genome assemblies of three plant species using optical mapping and chromosome conformation capture data. *Genome research*, 27(5):778–786, 2017.

[125] Ryan Tewhey, Vikas Bansal, Ali Torkamani, Eric J Topol, and Nicholas J Schork. The importance of phase information for human genomics. *Nature Reviews Genetics*, 12(3):215, 2011.

[126] Gustavo Glusman, Hannah C Cox, and Jared C Roach. Whole-genome haplotyping approaches and genomic medicine. *Genome medicine*, 6(9):73, 2014.

[127] Stephan Schiffels and Richard Durbin. Inferring human population size and separation history from multiple genome sequences. *Nature genetics*, 46(8):919, 2014.

[128] Matthew W Snyder, Andrew Adey, Jacob O Kitzman, and Jay Shendure. Haplotype-resolved genome sequencing: experimental methods and applications. *Nature Reviews Genetics*, 16(6):344, 2015.

[129] Sharon R Browning and Brian L Browning. Haplotype phasing: existing methods and new developments. *Nature Reviews Genetics*, 12(10):703, 2011.

[130] Peter Edge, Vineet Bafna, and Vikas Bansal. Hapcut2: robust and accurate haplotype assembly for diverse sequencing technologies. *Genome research*, pages gr–213462, 2016.

[131] Stefano Beretta, Murray D Patterson, Simone Zaccaria, Gianluca Della Vedova, and Paola Bonizzoni. Hapchat: Adaptive haplotype assembly for efficiently leveraging high coverage in long reads. *BMC bioinformatics*, 19(1):252, 2018.

[132] Jana Ebler, Marina Haukness, Trevor Pesout, Tobias Marschall, and Benedict Paten. Haplotype-aware genotyping from noisy long reads. *bioRxiv*, page 293944, 2018.

[133] Niranjan Nagarajan and Mihai Pop. Sequence assembly demystified. *Nature Reviews Genetics*, 14(3):157–167, 2013.

[134] Jason R Miller, Sergey Koren, and Granger Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315–327, 2010.

[135] Eugene W Myers. The fragment assembly string graph. *Bioinformatics*, 21(suppl 2):ii79–ii85, 2005.

[136] Niranjan Nagarajan and Mihai Pop. Parametric complexity of sequence assembly: theory and applications to next generation sequencing. *Journal of computational biology*, 16(7):897–908, 2009.

[137] J Craig Venter, Hamilton O Smith, and Leroy Hood. A new strategy for genome sequencing. *Nature*, 381(6581):364, 1996.

[138] Yang Dong, Min Xie, Yu Jiang, Nianqing Xiao, Xiaoyong Du, Wenguang Zhang, Gwenola Tosser-Klopp, Jinhuan Wang, Shuang Yang, Jie Liang, et al. Sequencing and automated whole-genome optical mapping of the genome of a domestic goat (capra hircus). *Nature biotechnology*, 31(2):135–141, 2013.

[139] Jennifer M Shelton, Michelle C Coleman, Nic Herndon, Nanyan Lu, Ernest T Lam, Thomas Anantharaman, Palak Sheth, and Susan J Brown. Tools and pipelines for bionano data: molecule assembly pipeline and fasta super scaffolding tool. *BMC genomics*, 16(1):734, 2015.

[140] Grace XY Zheng, Billy T Lau, Michael Schnall-Levin, Mirna Jarosz, John M Bell, Christopher M Hindson, Sofia Kyriazopoulou-Panagiotopoulou, Donald A Masquelier, Landon Merrill, Jessica M Terry, et al. Haplotyping germline and cancer genomes with high-throughput linked-read sequencing. *Nature biotechnology*, 2016.

[141] Sarah Yeo, Lauren Coombe, René L Warren, Justin Chu, and Inanç Birol. Arcs: Scaffolding genome drafts with linked reads. *Bioinformatics*, 2017.

[142] Marieke Simonis, Petra Klous, Erik Splinter, Yuri Moshkin, Rob Willemsen, Elzo De Wit, Bas Van Steensel, and Wouter De Laat. Nuclear organization of active and inactive chromatin domains uncovered by chromosome conformation capture–on-chip (4c). *Nature genetics*, 38(11):1348–1354, 2006.

[143] Joshua N Burton, Andrew Adey, et al. Chromosome-scale scaffolding of de novo genome assemblies based on chromatin interactions. *Nature biotechnology*, 31(12):1119–1125, 2013.

[144] Jesse R Dixon, Siddarth Selvaraj, Feng Yue, Audrey Kim, Yan Li, Yin Shen, Ming Hu, Jun S Liu, and Bing Ren. Topological domains in mammalian genomes identified by analysis of chromatin interactions. *Nature*, 485(7398):376–380, 2012.

[145] John Eid, Adrian Fehr, Jeremy Gray, Khai Luong, John Lyle, Geoff Otto, Paul Peluso, David Rank, Primo Baybayan, Brad Bettman, et al. Real-time dna sequencing from single polymerase molecules. *Science*, 323(5910):133–138, 2009.

[146] Heng Li. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, 32(14):2103–2110, 2016.

[147] Alec Wysoker, Kathleen Tibbetts, and Tim Fennell. Picard tools version 1.90, April 2013.

[148] Sergey Koren, Brian P Walenz, Konstantin Berlin, et al. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome research*, 27(5):722–736, 2017.

[149] Chen-Shan Chin, Paul Peluso, Fritz J. Sedlazeck, Maria Nattestad, Gregory T. Concepcion, Alicia Clum, Christopher Dunn, Ronan O'Malley, Rosa Figueroa-Balderas, Abraham Morales-Cruz, Grant R. Cramer, Massimo Delledonne, Chongyuan Luo, Joseph R. Ecker, Dario Cantu, David R. Rank, and Michael C. Schatz. Phased diploid genome assembly with single molecule real-time sequencing. *bioRxiv*, 2016.

[150] Mikhail Kolmogorov, Jeffrey Yuan, Yu Lin, and Pavel Pevzner. Assembly of long error-prone reads using repeat graphs. *bioRxiv*, 2018.

[151] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3):449–467, 1965.

[152] Matthias Poloczek and Mario Szegedy. Randomized greedy algorithms for the maximum matching problem with new analysis. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 708–717. IEEE, 2012.

[153] Valerie A Schneider, Tina Graves-Lindsay, Kerstin Howe, Nathan Bouk, Hsiu-Chuan Chen, Paul A Kitts, Terence D Murphy, Kim D Pruitt, Françoise Thibaud-Nissen, Derek Albracht, et al. Evaluation of grch38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome research*, 27(5):849–864, 2017.

[154] Miten Jain, Sergey Koren, Josh Quick, Arthur C Rand, et al. Nanopore sequencing and assembly of a human genome with ultra-long reads. *bioRxiv*, page 128835, 2017.

[155] Stefan Kurtz, Adam Phillippy, Arthur L Delcher, Michael Smoot, Martin Shumway, Corina Antonescu, and Steven L Salzberg. Versatile and open software for comparing large genomes. *Genome biology*, 5(2):R12, 2004.

[156] Nayanah Siva. 1000 genomes project, 2008.

[157] Steven L Salzberg, Adam M Phillippy, Aleksey Zimin, Daniela Puiu, Tanja Magoc, Sergey Koren, Todd J Treangen, Michael C Schatz, Arthur L Delcher, Michael Roberts, et al. Gage: A critical evaluation of genome assemblies and assembly algorithms. *Genome research*, 22(3):557–567, 2012.

[158] Francesco Vezzi, Giuseppe Narzisi, and Bud Mishra. Feature-by-feature–evaluating de novo sequence assembly. *PloS one*, 7(2):e31002, 2012.

[159] Natalia Naumova, Maxim Imakaev, Geoffrey Fudenberg, et al. Organization of the mitotic chromosome. *Science*, 342(6161):948–953, 2013.

[160] Kipper Fletez-Brant, Yunjiang Qiu, David U Gorkin, Ming Hu, and Kasper D Hansen. Removing unwanted variation between samples in hi-c experiments. *bioRxiv*, 2017.

[161] Klaus-Peter Koepfli, Benedict Paten, Genome 10K Community of Scientists, and Stephen J O?Brien. The genome 10k project: a way forward. *Annu. Rev. Anim. Biosci.*, 3(1):57–111, 2015.

[162] Nynke L Van Berkum, Erez Lieberman-Aiden, Louise Williams, Maxim Imakaev, Andreas Gnirke, Leonid A Mirny, Job Dekker, and Eric S Lander. Hi-c: a method to study the three-dimensional architecture of genomes. *Journal of visualized experiments: JoVE*, (39), 2010.

[163] Erik Garrison and Gabor Marth. Haplotype-based variant detection from short-read sequencing. *arXiv preprint arXiv:1207.3907*, 2012.

[164] Bruce J Walker, Thomas Abeel, Terrance Shea, Margaret Priest, Amr Abouelliel, Sharadha Sakthikumar, Christina A Cuomo, Qiandong Zeng, Jennifer Wortman, Sarah K Young, et al. Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. *PloS one*, 9(11):e112963, 2014.

[165] Justin Zook. Genome in a bottle. 2012.

[166] Sergey Koren and Adam M Phillippy. One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly. *Current opinion in microbiology*, 23:110–120, 2015.

[167] Sergey Koren, Todd J Treangen, and Mihai Pop. Bambus 2: scaffolding metagenomes. *Bioinformatics*, 27(21):2964–2971, 2011.

[168] Todd J Treangen, Sergey Koren, Daniel D Sommer, Bo Liu, Irina Astrovskaya, Brian Ondov, Aaron E Darling, Adam M Phillippy, and Mihai Pop. Metamos: a modular and open source metagenomic assembly and analysis pipeline. *Genome biology*, 14(1):R2, 2013.

[169] Song Gao, Wing-Kin Sung, and Niranjan Nagarajan. Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. *Journal of Computational Biology*, 18(11):1681–1691, 2011.

[170] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177, 2001.

[171] Kamesh Madduri, David Ediger, Karl Jiang, David A Bader, and Daniel Chavarria-Miranda. A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–8. IEEE, 2009.

[172] Robert Geisberger, Peter Sanders, and Dominik Schultes. Better approximation of betweenness centrality. In *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, pages 90–100. Society for Industrial and Applied Mathematics, 2008.

[173] Matteo Riondato and Evgenios M Kornaropoulos. Fast approximation of betweenness centrality through sampling. *Data Mining and Knowledge Discovery*, 30(2):438–475, 2016.

[174] Paul Medvedev, Konstantinos Georgiou, Gene Myers, and Michael Brudno. Computability of models for sequence assembly. In *International Workshop on Algorithms in Bioinformatics*, pages 289–301. Springer, 2007.

[175] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[176] Hubert W Lilliefors. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American statistical Association*, 62(318):399–402, 1967.

[177] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[178] Migun Shakya, Christopher Quince, James H Campbell, Zamin K Yang, Christopher W Schadt, and Mircea Podar. Comparative metagenomic and rrna microbial diversity characterization using archaeal and bacterial synthetic communities. *Environmental microbiology*, 15(6):1882–1899, 2013.

[179] NA Joshi, JN Fass, et al. Sickle: A sliding-window, adaptive, quality-based trimming tool for fastq files (version 1.33)[software], 2011.

[180] Lawrence Mitchell, Terence M Sloan, Muriel Mewissen, Peter Ghazal, Thorsten Forster, Michal Piotrowski, and Arthur S Trew. A parallel random forest classifier for r. In *Proceedings of the second international workshop on Emerging computational methods for the life sciences*, pages 1–6. ACM, 2011.

[181] Barbara A Methé, Karen E Nelson, Mihai Pop, Heather H Creasy, Michelle G Giglio, Curtis Huttenhower, Dirk Gevers, Joseph F Petrosino, Sahar Abubucker, Jonathan H Badger, et al. A framework for human microbiome research. *Nature*, 486(7402):215, 2012.

[182] Matthias Hess, Alexander Sczyrba, Rob Egan, Tae-Wan Kim, Harshal Chokhawala, Gary Schroth, Shujun Luo, Douglas S Clark, Feng Chen, Tao Zhang, et al. Metagenomic discovery of biomass-degrading genes and genomes from cow rumen. *Science*, 331(6016):463–467, 2011.

[183] Kenneth H Nealson and J Craig Venter. Metagenomics and the global ocean survey: what's in it for us, and why should we care? *The ISME Journal*, 1(3):185, 2007.

[184] Rachel Mackelprang, Mark P Waldrop, Kristen M DeAngelis, Maude M David, Krystle L Chavarria, Steven J Blazewicz, Edward M Rubin, and Janet K Jansson. Metagenomic analysis of a permafrost microbial community reveals a rapid response to thaw. *Nature*, 480(7377):368, 2011.

[185] Rolf Daniel. The metagenomics of soil. *Nature Reviews Microbiology*, 3(6):470, 2005.

[186] Dinghua Li, Chi-Man Liu, Ruibang Luo, Kunihiko Sadakane, and Tak-Wah Lam. Megahit: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de bruijn graph. *Bioinformatics*, 31(10):1674–1676, 2015.

[187] Toshiaki Namiki, Tsuyoshi Hachiya, Hideaki Tanaka, and Yasubumi Sakakibara. Metavelvet: an extension of velvet assembler to de novo metagenome assembly from short sequence reads. *Nucleic acids research*, 40(20):e155–e155, 2012.

[188] Sébastien Boisvert, Frédéric Raymond, Élénie Godzaridis, François Laviolette, and Jacques Corbeil. Ray meta: scalable de novo metagenome assembly and profiling. *Genome biology*, 13(12):R122, 2012.

[189] Bahlul Haider, Tae-Hyuk Ahn, Brian Bushnell, Juanjuan Chai, Alex Copeland, and Chongle Pan. Omega: an overlap-graph de novo assembler for metagenomics. *Bioinformatics*, 30(19):2717–2722, 2014.

[190] Howard Ochman, Jeffrey G Lawrence, and Eduardo A Groisman. Lateral gene transfer and the nature of bacterial innovation. *nature*, 405(6784):299, 2000.

[191] Pedro Gómez, Steve Paterson, Luc De Meester, Xuan Liu, Luca Lenzi, MD Sharma, Kerensa McElroy, and Angus Buckling. Local adaptation of a bacterium is as important as its presence in structuring a natural microbial community. *Nature communications*, 7:12453, 2016.

[192] Paul Igor Costea, Robin Munch, Luis Pedro Coelho, Lucas Paoli, Shinichi Sunagawa, and Peer Bork. metasnv: a tool for metagenomic strain level analysis. *PLoS One*, 12(7):e0182392, 2017.

[193] Stephen Nayfach, Beltran Rodriguez-Mueller, Nandita Garud, and Katherine S Pollard. An integrated metagenomics pipeline for strain profiling reveals novel patterns of bacterial transmission and biogeography. *Genome research*, 2016.

[194] Chengwei Luo, Rob Knight, Heli Siljander, Mikael Knip, Ramnik J Xavier, and Dirk Gevers. Constrains identifies microbial strains in metagenomic datasets. *Nature biotechnology*, 33(10):1045, 2015.

[195] Duy Tin Truong, Adrian Tett, Edoardo Pasolli, Curtis Huttenhower, and Nicola Segata. Microbial strain-level population structure and genetic diversity from metagenomes. *Genome research*, 2017.

[196] Zamin Iqbal, Mario Caccamo, Isaac Turner, Paul Flicek, and Gil McVean. De novo assembly and genotyping of variants using colored de bruijn graphs. *Nature genetics*, 44(2):226, 2012.

[197] Jurgen F Nijkamp, Mihai Pop, Marcel JT Reinders, and Dick de Ridder. Exploring variation-aware contig graphs for (comparative) metagenomics using marygold. *Bioinformatics*, 29(22):2826–2834, 2013.

[198] Yu Peng, Henry CM Leung, Siu-Ming Yiu, and Francis YL Chin. Idba–a practical iterative de bruijn graph de novo assembler. In *Annual international conference on research in computational molecular biology*, pages 426–440. Springer, 2010.

[199] Igor Mandric, Sergey Knyazev, Alex Zelikovsky, and Bonnie Berger. Repeat-aware evaluation of scaffolding tools. *Bioinformatics*, 1:8, 2018.

[200] Jay Ghurye and Mihai Pop. Better identification of repeats in metagenomic scaffolding. In *International Workshop on Algorithms in Bioinformatics*, pages 174–184. Springer, 2016.

[201] John Hopcroft and Robert Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973.

[202] Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with spqr-trees. *Algorithmica*, 15(4):302–318, 1996.

[203] Carsten Gutwenger and Petra Mutzel. A linear time implementation of spqr-trees. In *International Symposium on Graph Drawing*, pages 77–90. Springer, 2000.

[204] Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W Klau, Karsten Klein, and Petra Mutzel. The open graph drawing framework (ogdf). *Handbook of Graph Drawing and Visualization*, 2011:543–569, 2013.

[205] Zvi Galil, Silvio Micali, and Harold Gabow. An $O(EVlogV)$ algorithm for finding a maximal weighted matching in general graphs. *SIAM Journal on Computing*, 15(1):120–130, 1986.

[206] Lalena Wallace, Sean C Daugherty, Sushma Nagaraj, J Kristie Johnson, Anthony D Harris, and David A Rasko. The use of comparative genomics to characterize the diversity of acinetobacter baumannii surveillance isolates in a health care institution. *Antimicrobial agents and chemotherapy*, pages AAC–00477, 2016.

[207] Eliana De Gregorio, Giustina Silvestro, Mauro Petrillo, Maria Stella Carlomagno, and Pier Paolo Di Nocera. Enterobacterial repetitive intergenic consensus sequence repeats in yersiniae: genomic organization and functional properties. *Journal of bacteriology*, 187(23):7945–7954, 2005.

[208] Esther Singer, Bill Andreopoulos, Robert M Bowers, Janey Lee, Shweta Deshpande, Jennifer Chiniquy, Doina Ciobanu, Hans-Peter Klenk, Matthew Zane, Christopher Daum, et al. Next generation sequencing data of a defined microbial mock community. *Scientific data*, 3:160081, 2016.

[209] Alla Mikheenko, Vladislav Saveliev, and Alexey Gurevich. Metaquast: evaluation of metagenome assemblies. *Bioinformatics*, 32(7):1088–1090, 2015.

[210] Gerard Muyzer, Ellen C De Waal, and Andre G Uitterlinden. Profiling of complex microbial populations by denaturing gradient gel electrophoresis analysis of polymerase chain reaction-amplified genes coding for 16s rrna. *Applied and environmental microbiology*, 59(3):695–700, 1993.

[211] James R White, Saket Navlakha, Niranjan Nagarajan, Mohammad-Reza Ghodsi, Carl Kingsford, and Mihai Pop. Alignment and clustering of phylogenetic markers-implications for microbial diversity studies. *BMC bioinformatics*, 11(1):152, 2010.

[212] Lusheng Wang and Tao Jiang. On the complexity of multiple sequence alignment. *Journal of computational biology*, 1(4):337–348, 1994.

[213] J Gregory Caporaso, Christian L Lauber, William A Walters, Donna Berg-Lyons, James Huntley, Noah Fierer, Sarah M Owens, Jason Betley, Louise Fraser, Markus Bauer, et al. Ultra-high-throughput microbial community analysis on the illumina hiseq and miseq platforms. *The ISME journal*, 6(8):1621–1624, 2012.

[214] Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.

[215] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[216] Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[217] Eugene W Myers. An O (ND) difference algorithm and its variations. *Algorithmica*, 1(1):251–266, 1986.

[218] William J. Masek and Mike Paterson. A faster algorithm computing string edit distances. *J. Comput. Syst. Sci.*, 20(1):18–31, 1980.

[219] William J. Masek and Michael S. Paterson. How to compute string-edit distances quickly. pages 337–349, 1983.

[220] Gene Myers. A fast bit-vector algorithm for approximate string matching based on dynamic programming. *Journal of the ACM (JACM)*, 46(3):395–415, 1999.

[221] Dan Gusfield. *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge university press, 1997.

[222] Julie D Thompson, Toby Gibson, Des G Higgins, et al. Multiple sequence alignment using clustalw and clustalx. *Current protocols in bioinformatics*, pages 2–3, 2002.

[223] DOTREE Plotree and DOTGRAM Plotgram. Phylip-phylogeny inference package (version 3.2). *cladistics*, 5(163):6, 1989.

[224] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[225] Jens Roat Kultima, Shinichi Sunagawa, Junhua Li, Weineng Chen, Hua Chen, Daniel R Mende, Manimozhiyan Arumugam, Qi Pan, Binghang Liu, Junjie Qin, et al. Mocat: a metagenomics assembly and gene prediction toolkit. *PloS one*, 7(10):e47656, 2012.

[226] Peter J Turnbaugh, Ruth E Ley, Micah Hamady, Claire M Fraser-Liggett, Rob Knight, and Jeffrey I Gordon. The human microbiome project. *Nature*, 449(7164):804, 2007.

[227] Shinichi Sunagawa, Luis Pedro Coelho, Samuel Chaffron, Jens Roat Kultima, Karine Labadie, Guillem Salazar, Bardya Djahanschiri, Georg Zeller, Daniel R Mende, Adriana Alberti, et al. Structure and function of the global ocean microbiome. *Science*, 348(6237):1261359, 2015.

[228] Bertrand Routy, Emmanuelle Le Chatelier, Lisa Derosa, Connie PM Duong, Maryam Tidjani Alou, Romain Daillère, Aurélie Fluckiger, Meriem Messaoudene, Conrad Rauber, Maria P Roberti, et al. Gut microbiome influences efficacy of pd-1–based immunotherapy against epithelial tumors. *Science*, 359(6371):91–97, 2018.

[229] Kristoffer Forslund, Falk Hildebrand, Trine Nielsen, Gwen Falony, Emmanuelle Le Chatelier, Shinichi Sunagawa, Edi Prifti, Sara Vieira-Silva, Valborg Gudmundsdottir, Helle Krogh Pedersen, et al. Disentangling type 2 diabetes and metformin treatment signatures in the human gut microbiota. *Nature*, 528(7581):262, 2015.

[230] Junjie Qin, Yingrui Li, Zhiming Cai, Shenghui Li, Jianfeng Zhu, Fan Zhang, Suisha Liang, Wenwei Zhang, Yuanlin Guan, Dongqian Shen, et al. A metagenome-wide association study of gut microbiota in type 2 diabetes. *Nature*, 490(7418):55, 2012.

[231] Emmanuelle Le Chatelier, Trine Nielsen, Junjie Qin, Edi Prifti, Falk Hildebrand, Gwen Falony, Mathieu Almeida, Manimozhiyan Arumugam, Jean-Michel Batto, Sean Kennedy, et al. Richness of human gut microbiome correlates with metabolic markers. *Nature*, 500(7464):541, 2013.

[232] Alexandra Meziti, Despina Tsementzi, Konstantinos Ar. Kormas, Hera Karayanni, and Konstantinos T Konstantinidis. Anthropogenic effects on bacterial diversity and function along a river-to-estuary gradient in northwest greece revealed by metagenomics. *Environmental microbiology*, 18(12):4640–4652, 2016.

[233] Eugene W. Myers. An O(ND) difference algorithm and its variations. *Algorithmica*, 1(1):251–266, Nov 1986.

[234] Esko Ukkonen. Algorithms for approximate string matching. *Inf. Control*, 64(1-3):100–118, March 1985.

[235] Maxime Crochemore, Gad M. Landau, and Michal Ziv-Ukelson. A subquadratic sequence alignment algorithm for unrestricted scoring matrices. *SIAM J. Comput.*, 32(6):1654–1673, June 2003.

[236] Youngho Kim, Joong Chae Na, Heejin Park, and Jeong Seop Sim. A space-efficient alphabet-independent four-russians' lookup table and a multithreaded four-russians' edit distance algorithm. *Theor. Comput. Sci.*, 656:173–179, 2016.

[237] Ryan R. Wick, Mark B. Schultz, Justin Zobel, and Kathryn E. Holt. Bandage: interactive visualization of de novo genome assemblies. *Bioinformatics*, 31(20):3350, 2015.

[238] C. B. Nielsen, S. D. Jackman, I. Birol, and S. J. M. Jones. Abyss-explorer: Visualizing genome sequence assemblies. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):881–888, Nov 2009.

[239] Michael C Schatz, Adam M Phillippy, Ben Shneiderman, and Steven L Salzberg. Hawkeye: an interactive visual analytics tool for genome assemblies. *Genome biology*, 8(3):R34, 2007.

[240] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[241] Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Kiem phong Vo. A technique for drawing directed graphs. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 19(3):214–230, 1993.

[242] Kristi E Kim, Paul Peluso, Primo Babayan, P Jane Yeadon, Charles Yu, William W Fisher, Chen-Shan Chin, Nicole A Rapicavoli, David R Rank, Joachim Li, et al. Long-read, whole-genome shotgun sequence data for five model organisms. *Scientific data*, 1:140045, 2014.

[243] Konstantin Berlin, Sergey Koren, Chen-Shan Chin, James P Drake, Jane M Landolin, and Adam M Phillippy. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology*, 33(6):623–630, 2015.

[244] Sibren Isaacman, Richard Becker, Ramon Caceres, Margaret Martonosi, James Rowland, Alexander Varshavsky, and Walter Willinger. Human Mobility Modeling at Metropolitan Scales. *10th ACM International Conference on Mobile Systems, Applications and Services (MobiSys 2012)*, 2012.

[245] Richard Becker, Ramon Caceres, Karrie Hanson, Ji Loh, Simon Urbanek, Alexander Varshavsky, and Christopher Volinsky. A Tale of One City: Using Cellular Network Data for Urban Planning. *IEEE Pervasive Computing* , 2010.

[246] Vanessa Frias-Martinez, Cristina Soguero-Ruiz, Enrique Frias-Martinez, and Malvina Josephidou. Forecasting socioeconomic trends with cell phone records. In *Proceedings of the 3rd ACM Symposium on Computing for Development*, ACM DEV '13, 2013.

[247] R. Lambiotte, V Blondel, Kerchove C., Huens E., Prieur C., Smoreda Z., and P. Dooren. Geographical dispersal of mobile communications networks. *Physica A: Statistical Mechanics and its Applications*, 387(21):5317–5325, 2008.

[248] R. Ahas, A. Aasa, Y. Yuan, M. Raubal, Z. Smoreda, Y. Liu, C. Ziemlicki, M. Tiru, and M. Zook. Everyday spacetime geographies: using mobile phone-based sensor data to monitor urban activity in harbin, paris, and tallinn. *International Journal of Geographical Information Science*, 29(11):2017–2039, 2015.

[249] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, and Ryosuke Shibasaki. Intelligent system for urban emergency management during large-scale disaster. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[250] Yingrui Li, Hancheng Zheng, Ruibang Luo, Honglong Wu, Hongmei Zhu, Ruiqiang Li, Hongzhi Cao, Boxin Wu, Shujia Huang, Haojing Shao, et al. Structural variation in two human genomes mapped at single-nucleotide resolution by whole genome de novo assembly. *Nature biotechnology*, 29(8):723, 2011.

[251] Lars Feuk, Andrew R Carson, and Stephen W Scherer. Structural variation in the human genome. *Nature Reviews Genetics*, 7(2):85, 2006.

[252] Jonathan Sebat, B Lakshmi, Dheeraj Malhotra, Jennifer Troge, Christa Lese-Martin, Tom Walsh, Boris Yamrom, Seungtai Yoon, Alex Krasnitz, Jude Kendall, et al. Strong association of de novo copy number mutations with autism. *Science*, 2007.

[253] Michael J Morowitz, Vincent J Denef, Elizabeth K Costello, Brian C Thomas, Valeriy Poroyko, David A Relman, and Jillian F Banfield. Strain-resolved community genomic analysis of gut microbial colonization in a premature infant. *Proceedings of the National Academy of Sciences*, 108(3):1128–1133, 2011.

[254] Ian Lo, Vincent J Denef, Nathan C VerBerkmoes, Manesh B Shah, Daniela Goltsman, Genevieve DiBartolo, Gene W Tyson, Eric E Allen, Rachna J Ram, J Chris Detter, et al. Strain-resolved community proteomics reveals recombining genomes of acidophilic bacteria. *Nature*, 446(7135):537, 2007.

[255] Itai Sharon, Michael J Morowitz, Brian C Thomas, Elizabeth K Costello, David A Relman, and Jillian F Banfield. Time series community genomics analysis reveals rapid shifts in bacterial species, strains, and phage during infant gut colonization. *Genome research*, 23(1):111–120, 2013.