The Pennsylvania State University
The Graduate School
College of Engineering

ASSEMBLY ALGORITHMS FOR NEXT-GENERATION SEQUENCE DATA

A Dissertation in
Computer Science and Engineering
by
Aakrosh Ratan

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2009

The dissertation of Aakrosh Ratan was reviewed and approved* by the following:

Webb Miller
Professor of Biology and Computer Science and Engineering
Dissertation Adviser, Chair of Committee

Piotr Berman
Associate Professor of Computer Science and Engineering

Stephan Schuster
Professor of Biochemistry and Molecular Biology

Raj Acharya
Professor of Computer Science and Engineering
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

Next-generation sequencing is revolutionizing genomics, promising higher coverage at a lower cost per base when compared to Sanger sequencing. Shorter reads and higher error rates from these new instruments necessitate the development of new algorithms and software. This dissertation describes approaches to tackle some problems related to genome assembly with these short fragments.

We describe YASRA (Yet Another Short Read Assembler), that performs comparative assembly of short reads using a reference genome, which can differ substantially from the genome being sequenced. We explain the algorithm and present the results of assembling one ancient-mitochondrial and one plastid dataset. Comparing the performance of YASRA with the AMOScmp-shortReads and Newbler mapping assemblers (version 2.0.00.17) as template genomes are varied, we find that YASRA generates fewer contigs with higher coverage and fewer errors. We also analyze situations where the use of comparative assembly outperforms de novo assembly, and vice-versa, and compare the performance of YASRA with that of the Velvet (version 0.7.53) and Newbler de novo assemblers (version 2.0.00.17).

We utilize the concept of "overlap-graphs" from YASRA to find genetic differences within a target species. We describe a simple pipeline for deducing such locations of variation in the presence of a reference genome and then extend it to deduce polymorphisms in a species without the help of a reference genome. Our implementation of this algorithm, DIAL (De Novo Identification of Alleles) is described. The method works even when the coverage is insufficient for de novo assembly and can be extended to determine small indels (insertions/deletions). We evaluate the effectiveness of the approach using published Roche/454 sequence data of Dr. James Watson to detect heterozygous locations. We also apply our approach on recent Illumina data from Orangutan, in each case comparing our results to those from computational analysis that used a reference genome sequence.

Contents

List of Figures **vii**

List of Tables **viii**

Acknowledgments **ix**

1 Introduction **1**

1.1 Motivation 1

1.2 Sequencing Methods 2

1.2.1 Next-Generation Sequencing 3

1.2.2 Third-Generation Sequencing 4

1.3 A Brief Introduction to Genome Assembly 6

1.3.1 Mapping Assemblers 7

1.3.2 Comparative Assemblers 8

1.3.3 De Novo Assemblers 9

1.4 Discussion 11

2 YASRA **12**

2.1 Motivation 12

2.2 Results 13

2.2.1 Assembly of a Woolly Mammoth mtDNA 13

2.2.2 Assembly of the Finger Millet (*Eleusine coracana*) Plastid Genome 14

2.2.3 Effectiveness of Assisted Assembly as the Template is Varied 15

2.3 Methods 16

2.3.1 Layout Generation 16

2.3.2 Selection and Trimming of Reads 18

2.3.3 Multiple Alignment of Reads 18

2.3.4 Consensus Generation 19

2.3.5 Iterative Gap Closure . . . 19
2.3.6 Error Detection . . . 20
2.4 Discussion and Case Studies . . . 20

3 Alignment Clusters **22**
3.1 Clusters . . . 22
3.2 Clusters with a Reference . . . 23
3.2.1 Methods . . . 24
3.2.2 Case Studies . . . 24
3.3 Clusters without a Reference . . . 26
3.3.1 Methods . . . 26
3.3.2 Case Studies . . . 27

4 DIAL **28**
4.1 Results . . . 28
4.1.1 Overview of the Computational Problem . . . 29
4.1.2 Watson Genome Dataset . . . 32
4.1.3 Orangutan Dataset . . . 35
4.1.4 Mule Deer Transcriptome Dataset . . . 36
4.2 Methods . . . 38
4.2.1 Masking of Repetitive Sequences . . . 38
4.2.2 Calculation of Overlaps . . . 40
4.2.3 Determination of SNPs . . . 41
4.3 Case Studies . . . 42

5 Conclusions and Future Work **44**
5.1 Hybrid Assemblies . . . 44
5.2 YASRA . . . 46
5.3 Assembly Validation . . . 47

Bibliography **48**

Appendix A. Assembly Paradigms **59**
A.1 Shortest Superstring . . . 59
A.2 Overlap-Layout-Consensus . . . 60
A.3 Sequencing by Hybridization . . . 61
A.4 Hierarchical Assembly . . . 62
A.5 Machine Learning . . . 63

Appendix B. Mapping Assemblers **65**

B.1 Indexing of Reads with Hash Tables 65

B.2 Indexing of Reference Genomes with Hash Tables 66

B.3 Indexing of Reference Genomes with BWT/Suffix Array 67

List of Figures

2.1 YASRA pipeline 17
2.2 Division of reads into contained and non-contained 19
3.1 Example of a cluster data structure 23
4.1 Variation with Coverage (Watson genome dataset) 33
4.2 Magnification of the data in Figure 4.1 for 1-fold coverage or less. 34
4.3 Rates of erroneous SNP calls (Watson genome dataset) 35
4.4 Variation of SNPs with Coverage (Orangutan dataset) 37
4.5 DIAL pipeline 39

List of Tables

1.1 Comparison of next-generation sequencing technologies 5
2.1 Comparison of various assemblers for the M13 dataset 14
2.2 Comparison of various assemblers for the finger miller dataset 15
2.3 Comparison of various assemblers as the template is varied 16
2.4 LASTZ parameters for YASRA 17
3.1 SNP calls for human individuals 25
4.1 Theoretical analysis (Watson genome dataset) 31

Acknowledgments

I would like to take this opportunity to thank my advisor Webb Miller. He has introduced me to challenging problems, and guided me with constructive suggestions and feedback throughout this endeavor. His passion for research and work-ethic is both admirable and infectious. I am and will always be grateful for his substantial support.

I would also like to thank the other members of my current committee for their continued encouragement and feedback. Stephan has always been kind enough to include me in many interesting projects and his vision has heavily influenced the direction of my research. Besides that, I have enjoyed many discussions with colleagues in the lab: Bob Harris, Richard Burhans, Cathy Riemer, Belinda Giardine, Daniela Drautz, Ji Qi to name a few. I thank them all for their help and suggestions.

This dissertation draws heavily from two projects and the resulting manuscripts that were a part of my Ph.D. research and I would like to thank all the collaborators - this work would not have been possible without their efforts. Lastly, I would like to express my heartfelt gratitude to my parents for their unconditional love and support. I dedicate this thesis to them.

Chapter 1

Introduction

1.1 Motivation

The knowledge of sequences has become an essential part of every biological study today. The success of several whole-genome sequencing projects has spurred the field of comparative genomics, in which genomes from many species are compared to each other, leading to a better understanding and appreciation of the process of evolution. A complementary approach has involved studying genotypes of several individuals of the same species to understand the variability within the species, with the goal of using the genotype information to hopefully explain the phenotypes.

The new sequencing technologies are proving to be an ideal vehicle to achieve these goals, by producing data at a higher throughput and a lower cost than Sanger sequencing. Many research scientists are proceeding with projects to sequence a wide range of species [1] and individuals [2]. Large sequencing centers like the Washington University's Genome Center are producing 500 times the data that they were capable of in 2006 [3]. However, these next-generation sequencing technologies come with their own set of biases and sources of error. The reads generated using them are smaller in length and have a higher error rate than Sanger reads; nevertheless the high throughput and lower cost per base does makes them ideal for resequencing projects, where mapping the reads onto a pre-existing reference sequence helps to mitigate the problems.

Even though traditional software tools like BLAST [4] and BLAT [5] can handle short read data, a new set of tools is required to ensure a speedy turnaround for these ambitious projects. These programs need to be able to handle huge volumes of data with limited resources and they need to accommodate the error profile of these reads to ensure that the data is not misinterpreted. New software for traditional applications like genome/target resequencing and de novo assembly needs to be developed along with tools for novel applications like ChIP-Seq, transcriptome analysis, small

RNA analysis, methylation analysis and gene-expression analysis to ensure that the utility of these reads is maximized.

This dissertation describes some of the software and algorithms that have been developed by the author for sequences generated using the next-generation sequencing instruments. We describe and evaluate YASRA (Yet Another Short Read Assembler), a tool designed to bridge the gap between mapping assemblers (designed for resequencing) and de novo assemblers. YASRA performs comparative assembly of short reads using a reference genome, which can differ substantially from the genome being sequenced. We also describe the optimizations that were made to handle a large number of sequences efficiently. The primary step in all assemblers is the calculation of overlaps between the reads, either explicitly or implicitly, and this information is traditionally stored in an "overlap graph". The nodes in this graph represent the reads, and two nodes are connected by an edge if they are involved in a "proper" overlap. We formally introduce the concept of a "cluster", which is characterized by a node and its edges in the overlap graph. We describe a pipeline to deduce and assemble such clusters with a reference sequence. The idea of clusters is then extended to determine locations of variation in a species for which we lack a reference sequence. Variations include SNPs, insertions, deletions and other large-scale operations. A SNP (single-nucleotide polymorphism) is a DNA sequence variation when a single nucleotide in a genome differs between the different members of the same species, or between the paired copies of the same individual. The different nucleotides found at that location can be referred to as "alleles". We formalize the computational problem and describe our approach to selecting high quality SNPs for genotyping assays. The implementation of this approach, DIAL (De Novo Identification of Alleles) is evaluated on several datasets. The description of each tool is followed by case studies where it is used to answer some interesting biological questions.

We begin with a short description of the various sequencing methods, followed by an introduction to genome assembly. We briefly describe some of the current tools, and discuss the need for changes and enhancements.

1.2 Sequencing Methods

DNA sequencing utilizes biochemical methods to determine the order of nucleotide bases in a DNA oligonucleotide. The sequencing landscape has long been dominated by chain-termination methods developed by Frederick Sanger [6]. A number of large-scale sequencing projects have been accomplished by the cloning of DNA fragments into bacterial vectors, then amplification and purification of individual templates followed by Sanger sequencing. Sanger sequencing is capable of producing random sequence reads that are between 500 and 1000 bp long with an error rate of less than 1% at a cost of less than $0.001 per base.

The classical chain-termination method involves the use of modified dideoxynucleotides which cause termination of strand elongation during replication. The DNA sample is treated with a low concentration of chain terminating nucleotides in 4 separate lanes, one each for A,C,G,T resulting in DNA strands of varying length. These DNA strands are then heat-denatured and separated on a polyacrylamide-urea gel. Several variants of this basic principle are used. In one of the variations, the DNA fragments are tagged with nucleotides containing radioactive phosphorous for radio-labelling. Such fragments can be read by an optical system, facilitating faster and more economical analysis. These variants are called “dye-primer sequencing”.

One of the crowning achievements of Sanger sequencing was the Human Genome Project, which started in October 1990, and took more than 10 years and more than $300 million to complete. Even though it was a remarkable scientific achievement, it was immediately clear that the cost, time and energy expended in generating this reference sequence could not be scaled to sequence the thousands of genomes, that would be needed to understand human diversity. The US National Human Genome Research Institute (NHGRI) announced in October 2004 that it had awarded more than $38 million in grants to develop methods that would lower the cost of DNA sequencing [7]. The goal was to lower the cost of a complete sequence to $100,000 by 2009 and to $1000 by 2014. This decrease in cost would enable the dream of “personalized medicine”, where it would become routine to have the genome sequenced in an annual physical examination. This has led to a whole new crop of sequencing technologies collectively known as “next-generation sequencing technologies”. They have lived up to expectation, lowering the cost of sequencing a human-sized genome to about $48,000 [8] for a coverage of about 30X.

1.2.1 Next-Generation Sequencing

Most of the next-generation sequencing methods eliminate the bacterial cloning step, replacing it with amplification of single isolated DNA molecules. Some of the cloning bias issues can thus be avoided, and this has already led to closure of some gaps in the human reference sequence [9]. A comparison of the three main next-generation technologies is detailed in Table 1.1.

In the case of Roche/454 sequencing [10], a single strand of DNA is attached to a glass bead and amplified in a water-in-oil emulsion. The result is a separate emPCR (emulsion polymerase chain reaction) reaction and clones for the strand. The bead is then mixed with DNA polymerase (polymerase “reads” a DNA strand as a template and uses it to synthesize a new strand) and put in a “well”. Nucleotides then flow sequentially (in a specified order) onto it and as each base is added to form the complementary strand, the phosphorylation reaction is captured by an optical detector. Millions of such beads are incorporated in a single run, leading to a high throughput for the system. There is nothing to stop incorporation of multiple nucleotides of the same base at once, and hence the largest source of error in 454 reads is indels (insertions/deletions) caused by

erroneous homopolymer length calls.

Illumina/Solexa sequencing [11] starts with single stranded, adaptor oligo-ligated DNA fragments. These are attached to the surface of a glass flow-cell. Each flow-cell is divided into 8 lanes, which have covalently attached oligos complementary to specific adaptors ligated to the library fragments. An isothermal polymerase is used to amplify the clusters of sequences on the surface, which are then supplied with polymerase and four differentially placed fluorescent nucleotides. The nucleotides have an inactive $3'$-hydroxyl group which ensures that only one nucleotide is incorporated in each cycle. This is followed by an imaging step that attempts to identify the incorporated nucleotide, thus avoiding the homopolymer issue that plagues 454 reads. The major source of error for this technology is the wrong identification of the incorporated nucleotide.

ABI/SOLiD sequencing starts with oligo adaptor-linked DNA and amplifies each bead of DNA by emPCR. These beads are then covalently placed on a glass slide in the sequencer. Sequencing primers complementary to the adaptors are annealed on the surface, and a set of semi-degenerate 8mers and DNA ligase (which can link together two DNA strands that have double-strand break) is supplied to the mix. When an 8mer hybridizes, the ligase is used to seal the phosphate backbone. This is followed by an imaging step which tries to read the second or the fifth base on the 8mer, depending on the cycle number. A subsequent reaction is used to cleave the bases beyond the fifth base of the 8mer, removing the fluorescent group. The process occurs in steps that identify the sequence of each fragment in intervals of five nucleotides. The next cycle starts with the hybridization of a $n - 1$ position universal primer, thus permitting generation of millions of reads in a single run. Even though this technology aims to reduce the error rates by enabling "2-base encoding" which relies on the fact that the sequences of the 8mer probe are known, the error rate is still high. A drawback of the 2-base encoding is that areas in the genome with a higher SNP density get a lower coverage, or in some cases even none.

These next-generation technologies have accomplished what they set out to achieve and lowered the cost sequencing, bringing it within the reach of most scientists. The "third-generation" promises "single-molecule sequencing" and an improvement in the read lengths and accuracy, which were sacrificed to lower the costs. Some of these are just improvements to similar technology, while others are based on radically novel concepts.

1.2.2 Third-Generation Sequencing

Major players in this market include Helicos BioSciences, Complete Genomics, Pacific BioSciences and Oxford Nanopore Technologies. Here we give a brief description of each of these offerings. The Helicos Genetic Analysis System was recently used to sequence the human genome of Dr. S. Quake [12]. The system generates reads on average 32 bp long, with some reads exceeding 50 bp in length. These reads can cover 90% of the human genome with an error rate of about 3.2%,

Table 1.1: Comparison of next-generation sequencing technologies.

	Roche/454 FLX	**Illumina/Solexa**	**ABI/SOLiD**
Commercial release	2005	January 2007	October 2007
Sequencing Approach	pyrosequencing	polymerase based SBS	ligation
Sources of errors	indels	mismatches	mismatches
Average read length(bp)	450	76	50
Paired-end separation	2 kbp	200 bp and 2 kbp	600 bp - 10 kbp
Total yield/run	400 Mb	20 Gb	300 Gb
Cost/Mb($)	30	0.8	0.2
Starting material needed(μg)	1-5	3-5	3-5
Time/run	8 hours	10 days	10 days

which is much higher than that for the second generation systems. The drawbacks of the system include lack of a paired-end protocol, short reads, and a high error rate. Unfortunately, unlike most other contenders in this category, it hasn't really lowered the cost of sequencing any further.

Complete Genomics employs high-density DNA nanoarrays (glass slides) populated with DNA nano-balls, and uses a non-sequential unchained read technology called combinatorial probe-anchor ligation or cPAL to do the sequencing. Complete Genomics does not plan to sell sequencing machines to their clients, but instead to perform all of the work in-house. Their unique business model focuses on large-scale human genomics, thus reducing the cost associated with the service-based approach. They have sequenced a Caucasian HapMap sample NA07022 generating 91X average read coverage. They called about 3.3 million SNPs with a high concordance to the HapMap calls for the sample. They plan to sequence another 1000 genomes by the end of 2009 and 20,000 genomes in 2010. The company has said that they will soon start charging $20,000 per genome for an order of eight genomes or more, and $5,000 apiece for an order of 1,000 or more-with variable pricing in between.

Pacific Biosciences uses a single-molecule technology with DNA polymerase, that reads out the product of the sequencing reaction as it progresses. This Single Molecule Real Time (SMRT) technology is capable of read lengths of 600 bp on average with some maximum read lengths of about 3000 bp. The long reads make this technology exciting and they hope to have a commercial offering by the end of 2010.

Oxford Nanopore's nanopore sequencing [13] is the most intriguing of them all, even though it is not yet ready for commercial deployment. The technology promises to be the cheapest, while using no labeling, no expensive fluorescent reagents and no optical systems. It uses an adapted protein nanopore (small hole in an electrically insulating membrane) coupled with a processive exonuclease enzyme to sequence DNA. The enzyme cleaves individual bases from a strand of DNA, and sequentially introduces them into the aperture of the nanopore. An ionic current is con-

tinually flowing through the nanopore, and as individual bases travel through it, each creates a characteristic disruption in this current (since each base has a characteristic dwell time). This signal is recorded electronically and interpreted to identify the nucleotide. Recordings from multiple channels in parallel allows high-throughput sequencing of DNA. They recently signed a distribution contract with Illumina to exclusively market, sell, and distribute their sequencing products. However, no release date is set for their product.

1.3 A Brief Introduction to Genome Assembly

Genome assembly can be defined as the problem of finding the correct linear DNA sequence given a set of reads. Typically these reads result from shotgun sequencing (sequences are subdivided into smaller fragments, and subsequently re-assembled) genomic DNA or gene transcripts (ESTs). The problem can be broken down into two sub-problems: **layout**, where the correct placement of a read on the genome is inferred, and **consensus**, where the consensus sequence is generated from the multiple alignment of the reads. This formulation of the problem suffers from an immediate issue: that of repeats. The correct placement of reads with repeats cannot be guaranteed. The assembly problem should therefore be posed as the following [14]:

> Given a collection of reads with known DNA sequence together with additional constraints on their placements, find the DNA sequence of the original molecule.

The phrase “additional constraints” includes all kinds of ancillary information that can be used to aid in the correct placement of a read. This may include constraints on the global placement of the read, such as an alignment to a reference genome or a physical map data. It may also include the limitations on the relative placement of the reads such as the mate-pair distances and the overlap information. Besides these, some assemblers rely on quality values and other statistical assumptions to favor one candidate assembly over the other.

First generation assemblers began appearing in the late 1980’s and early 1990’s, piecing together the vast quantities of data generated by the automated sequencing machines. Most of the early assemblers [15, 16, 17, 18] were based on a simple theoretical model: the shortest superstring problem [19]. Others were based on paradigms like overlap-layout-consensus, which essentially breaks the assembly problem into three sub-problems. This approach is used by the Celera [20] and ARACHNE [21] assemblers which are probably the most successful assemblers for data generated using Sanger sequencing. Another novel approach to this problem, is based on “sequencing by hybridization” experiments [22]. Idury and Waterman proposed [23] it as an alternative to the OLC paradigm; the idea is to convert the reads into k-tuples and then treat it as an sequencing by hybridization experiment. Other notable attempts at solving the assembly

problem included the use of hierarchical assembly [24], where the DNA is broken into a collection of large fragments, which are then individually sequenced and assembled. A detailed description of these paradigms can be found in Appendix A.

Although some of the older assemblers like Celera and ARACHNE have been modified to handle short reads, the huge amount of data coupled with the high error rate from the new technologies dictated the development of new tools. Since most of the market for short reads is driven by resequencing, most of the tools for handling these reads have been designed with this application in mind. These assemblers use the existing reference sequence as a template, mapping the reads onto it, while allowing a certain number of mismatches. Some of them allow for gaps due to insertion/deletions as well, albeit that is achieved at a cost of speed. We discuss in brief, approaches adopted by assemblers for these short-reads. We also list some of the popular short-read mapping assemblers in Appendix B.

1.3.1 Mapping Assemblers

The mapping programs for next-generation sequencing can be divided into a few categories based on their approach to aligning the reads onto a reference sequence.

Indexing of Reads with Hash Tables

The first of these aligners was Eland, which is part of the software shipped by Illumina with their sequencing machines. Most of the other aligners in this category are heavily influenced by it. Eland builds multiple hash tables to index the reads and then scans the reference sequence against the hash tables to find the hits. The default behavior is to build six hash tables, which ensures that a sequence with less than or equal to two mismatches in the first 28 bp is always a hit [25]. The six hash tables correspond to six non-contiguous seed templates [26]. By default, these aligners index the first few bases of a read, which normally are the bases with the highest quality scores. The limitation of these aligners is the overhead of scanning the whole genome when only a few reads need to be aligned.

Indexing of Reference Genomes with Hash Tables

The aligners in this category work on similar principles as those in the above category, but they choose to index the reference genome instead of the reads. This way each read can be treated in succession and its hits can be output and processed, making the downstream pipeline simpler. However an increased memory footprint is required to build the index for large e.g. mammalian genomes.

Indexing of Reference Genomes with BWT/Suffix Array

Most of the aligners in this category use backward search [27] in conjunction with BWT (Burrows Wheeler Transform) [28], mimicking the top-down traversal of the prefix trie for the genome. This method can count the number of exact hits for a string of length m in $O(m)$ time, independent of the size of the genome. The exact repeats are collapsed on one path on the prefix trie, hence the reads are not aligned to them multiple times. This is the main reason why these aligners are faster when compared to aligners in the other categories. Most of the programs discussed above put a hard limit on the number of mismatches between the reads and the reference to achieve speed, but the aligners in this category are more flexible in that regard too.

1.3.2 Comparative Assemblers

There has been a tremendous increase in the number of genomes that have been sequenced over the past few years. This has prompted an interest in a comparative genome assembler, which uses a "reference" in order to guide the assembly of the target organism. In some cases the template or reference is a closely related organism whereas in other cases, it could be a different strain of the same genus or a different assembly of the same genome. Therefore they can be regarded as a bridge between the mapping and de novo assemblers. The scientific community has long recognized the benefit of assembling two closely related species or strains. The prominent examples of this has been in biodefense, where multiple strains of *Bacillus anthracis* have been sequenced to create genotyping information that can be used for forensic purposes [29]. Other examples include pyrosequencing of ancient mitochondria [30] and study of E. Coli to allow observation of bacterial evolution on a laboratory timescale [31].

Most of these efforts have used Comparative Genome Sequencing (CGS) technology in microarrays, but some of these [32] have used assemblers which use the template to find prospective positions for a read, thus creating a tiling map or a guide for the assembler. In AMOScmp [32], the overlap step is skipped entirely. The reads are aligned to the reference genome using a modified version of the MUMmer algorithm [33]. The remaining assembly stages – consensus generation and scaffolding – remain the same as in traditional assemblers.

The algorithm employed by AMOScmp to handle the differences between the template and the target can be enumerated as below.

1. **Read alignment.** The reads are aligned to the reference genome using MUMmer (which uses suffix trees to search for matches) and the reads with more than one placement are separated. The placement of the remaining reads is stored.

2. **Repeat resolution.** For each of the reads that cannot be placed unambiguously, the mate

pair information is used to disambiguate the placement. If that is not possible, then one of the locations is chosen at random.

3. **Layout refinement.** Indels and rearrangements between the target and reference complicate the mapping of the reads to the reference. In case of insertions and divergent DNA, AMOScmp breaks the assembly and uses a traditional assembly algorithm to construct the inserted segment. Mate pair information is used later to position and orient this segment into contigs. Variety of cases of rearrangement in either the target or the reference genome are handled by identifying their signatures.

4. **Consensus generation.** As in traditional assemblers, the consensus nucleotide sequence is generated by a multiple alignment of the reads which have been placed.

5. **Scaffolding.** The Bambus package [34] is used to determine the order and orient the contigs as determined by the mate-pair information.

1.3.3 De Novo Assemblers

De novo assembly of mammalian genomes using short reads remains an elusive goal, despite recent strides in assembling bacterial genomes. The short length of the reads and the volume of data both hinder the traditional approach of using an overlap graph for de novo assembly. The various approaches to de novo assembly with illumina/SOLiD reads can be divided into two broad categories: **greedy extensions** and **graph-based**. We briefly discuss these two approaches and the popular tools based on them.

Greedy Extensions

Most of the earliest assemblers for short reads were based on this paradigm. For example SHARCGS [35] uses a prefix tree to look up potential extensions for the reads. SSAKE [36] searches through a prefix tree to find the longest possible overlap between any two sequences, which are then assembled in a greedy manner. VCAKE [37] is very similar to SSAKE but instead of using a greedy extension of seeds, it considers all reads overlapping with the seed sequence. VCAKE extends the seed sequence one base at a time using the most commonly represented base from these matching reads, provided that the set of reads passes certain conditions and thresholds. Another tool QSRA [38] builds upon the VCAKE algorithm to make it base quality-aware. Its authors found that a quality aware approach led to longer contigs than those formed by VCAKE.

Graph Based

Newer tools for short reads are mostly based on more accurate graph based approaches. Edena [39] is based on the traditional overlap-layout-consensus approach. The assembler tries to overcome the high error rate of the short reads with over-sampling. In order to handle the high volume of short read data, it limits the overlaps to exact matches, which can be calculated very quickly using indexes. It also includes an error-removal step, where it resolves bubbles in the graph to remove spurious and transitive edges.

All of the other assemblers in this category use a de Bruijn graph approach to tackle the assembly problem. The fundamental data structure in the de Bruijn graph is based on k-mers, not reads, thus high redundancy is naturally handled by the graph without affecting the number of nodes. In addition, each repeat is present only once in the graph with explicit links to the different start and end points.

In Velvet [40] the formation of the de Bruijn graph is followed by a simplification step that removes transitive edges. This is followed by an error-correction step that uses topological features to identify genuine errors. Erroneous data create three types of structures: “tips” due to errors at the edges of reads, “bulges” due to internal read errors or to nearby tips connecting, and erroneous connections due to cloning errors or to distant merging tips. The three features are removed consecutively, followed by repeat resolution using the paired end reads to form the final assembly.

In ALLPATHS [41] the first step in the assembly involves error correction based on the k-mer counts. Very low k-mer counts result from incorrect k-mers (assuming the errors are random) and they are edited to make the read “correct”. This is followed by construction of a compact searchable data structure similar to the de Bruijn graph. Approximate unipaths (maximal unbranched sequence in a genome G, relative to a given minimum overlap K) can then be computed from this data structure, roughly by walking along the reads until a branch is encountered.

Euler-SR [42] follows a similar approach to the problem. The first step is “error correction”, similar to the analogous step in ALLPATHS. This is followed by maximum branching optimizations on the de Bruijn graph generated from the corrected reads, similar to that in EULER [43] and Velvet. The final step resolves repeats shorter than the length of the read. Some other popular assemblers include ABySS [44] , which implements a distributed representation of de Bruijn graphs to assemble the reads in parallel. The algorithm is similar to that used by Velvet and it was recently used to create a de novo assembly of a Yoruban male with some success. Another assembler, Taipan [45], uses a hybrid approach where it uses greedy extensions for contig construction, but at each step realizes enough of the corresponding read graph to make better decisions as to how assembly should continue.

1.4 Discussion

Assembly is a well studied problem, but it has experienced a resurgence due to the new challenges created by the development of the next-generation sequencing instruments. The de novo assemblers face the challenges of short reads and high error rates whereas the mapping assemblers are limited in their ability to handle high rates of polymorphism between the target and the template. This is where YASRA is different from the existing tools. It uses a genome which diverged upto a few million years ago and tries to use it to aid the assembly of the target species. It is similar to AMOScmp in this manner. AMOScmp did not handle short-reads until recently. Now AMOScmp-shortReads (a script which sets up parameters amenable to short reads) handles 454 and Illumina data, but the default parameters do not perform well if the reference is less than 90% identical to the reference. Divergent templates also leads to a fragmented assembly in the absence of mate-pair reads, something that YASRA improves on. It is also extremely difficult to tweak the parameters for a new dataset or if the read length of a particular technology changes. We compare YASRA to AMOScmp-shortReads and demonstrate that it performs better on a number of datasets and under a number of different conditions. We also provide several score-sets for different percent identity thresholds and read lengths to make tuning easier. Chapter 2 discusses the details of YASRA and evaluates its performance on several datasets.

These assembled genomes are commonly used to find the locations of variation in the target species. Even though this is a post-assembly process, it is hampered by the same issues that complicate assembly. DIAL, developed by the author tries to overcome those shortcomings by concentrating on the sub-problems, that are bound to give the most interesting results. DIAL creates an overlap graph and uses the expected coverage to find and ignore the reads that have a high probability of being from a repeat sub-structure. The remaining nodes in the graph can then be analyzed to find the polymorphisms. We discuss some preliminary approaches in Chapter 3 and then discuss DIAL in greater detail in Chapter 4. We also discuss steps taken to select a subset of these SNPs that are suitable for genotyping assays and its utility in conservation genomics.

Chapter 2

YASRA

We commented on the current state of mapping assemblers in Chapter 1. Most of these assemblers sacrifice sensitivity for speed and are designed for organisms with a low polymorphic rate. They normally use the genome from the same species as the template to assemble the target species. Comparative assemblers on the other hand, can handle reference sequences separated by millions of years from the target genome. There are limitations to their utility, but we believe they can play an important role in a significant proportion of assembly problems that arise with next-generation sequencing data. In this chapter, we describe a comparative assembler YASRA (Yet Another Short Read Assembler), developed by the author.

2.1 Motivation

The read lengths resulting from the next-generation sequencing technologies vary from 35-50 bp from Illumina/Solexa to 450 bp for the latest offering from Roche/454. These short reads must then be assembled to reconstruct the genome of the target organism. The first step in a de novo assembly involves calculation of overlaps among the reads. This is computationally the most expensive component of the assembly process. The difficulty is exacerbated for the high-throughput technologies since there are many putative overlaps, most of which turn out to be false. A comparative assembler takes an interesting approach to tackle this problem. It compares the reads with the sequence from an existing assembled genome to find the approximate placement for each read. This is implicitly the overlap information, which can then be used to form a multiple alignment and infer a consensus sequence. The strength of this approach lies in treating the sequenced genomes as valuable resources to aid the assembler. We believe this will become invaluable as more and more genomes get sequenced.

Most assemblers have an option to include a reference genome, but they take a conservative

approach to utilizing it. Most of them just use it to generate scaffolds that guide the finishing process. They also require adjustment of a lot of parameters to work satisfactorily on data from a new species. We propose a tool YASRA, which can outperform existing tools under certain circumstances, and requires rather limited tuning to work on a new dataset.

2.2 Results

We have applied and evaluated YASRA on a number of datasets. Here we report the results from assembling an ancient mtDNA dataset generated using 454 technology and another plastid genome dataset generated using Solexa technology. We compare the results from YASRA to that from Newbler (version 2.0.00.17), AMOScmp-shortReads and Velvet (version 0.7.53) assemblers. We then comment on the effectiveness of assisted assembly as the reference sequence is varied and highlight some of the limitations of our approach.

2.2.1 Assembly of a Woolly Mammoth (*Mammuthus primigenius*) Mitochondrial Genome

We published mitochondrial (mtDNA) genomes of several woolly mammoths which varied substantially in the amount of post-mortem DNA damage [30]. Here we use the most challenging of them, a specimen called M13, where the dataset consists of 952,792 Roche/454 reads with a mean length of only 50 bp. The resulting consensus sequence, is in Genbank [Genbank:EU153445.1]. It was derived using Newbler, CAP3W (an unpublished tool) and the low confidence regions were confirmed using PCR. The non-hypervariable region from an earlier submitted mammoth mtDNA sequence [46], which differs from the M13 assembly by 20 bp, was selected as the reference genome. For this experiment, all assemblers were run with the default parameters and arguments, since the template and the target genome are the same species.

Newbler and Velvet were used to generate de novo assemblies, and the resultant contigs were aligned with the M13 mtDNA using LASTZ [47] to find the relevant contigs (Only 0.62% of the reads are mtDNA). These contigs were then compared to EU153445.1; the results are detailed in Table 2.1. YASRA, AMOScmp-shortReads and Newbler mapping assembler generated contigs that covered 99.93%, 99.93% and 99.87% of the target genome. The assembly generated by YASRA, however, has fewer errors (6) compared to the assemblies generated by AMOScmp (7) and Newbler (8). YASRA was the only assembler to produce a single contig for the assembly. The de novo assembly by Newbler produced 2 contigs covering 99.83% of the target genome and produces more errors (16) than YASRA. Velvet, run with a k-mer value of 21 and a coverage cutoff of 3.0, produced 7 contigs covering 99.23% of the target genome but has many fewer errors (6) compared to Newbler.

Table 2.1: Comparison of various assemblers for the M13 mtDNA genome. We report the number of contigs, the number of nucleotide matches to the correct sequence, the number of mismatches, the number of nucleotides missing in the assembled sequence, number of nucleotides extra in the assembled sequence and time taken for the assembly. The score set used for YASRA is shown next to the name of the assembler.

Assembler	Contigs	Matches	Mismatches	Missing	Extra	Time(s)
Velvet	7	16,333	1	4	1	202
Newbler de novo	2	16,432	14	2	0	20,242
Newbler mapping	2	16,438	0	1	7	14
AMOScmp-shortReads	2	16445	0	4	3	169
YASRA(yasra95)	1	16,444	5	0	1	24

Deeper analysis of the assembled sequences reveals that Velvet generated assembly had a contig break in a region of around 100 bp spanned by very few reads, which resulted in its lower coverage on the reference. When compared to Newbler, YASRA performed better in homopolymer regions, probably as a result of the realignment step (See Methods).

2.2.2 Assembly of the Finger Millet (*Eleusine coracana*) Plastid Genome

Reads generated using Solexa technology are much shorter than the 454 generated reads. We assembled a Solexa dataset for the plastid genome of *Eleusine coracana* using the Sorghum plastid sequence [Genbank:NC_008602], as the reference. The reference is 95.8% identical to the target genome, as calculated later from the various reads which aligned to the reference. The Solexa dataset consists of 2,399,691 reads, each 31 bp, giving an approximate raw coverage of 300X of the plastid genome.

The results of the assembly and the differences between the reads and the assembled sequences are described in Table 2.2. The reads used by the various assemblers were extracted and realigned to the assembled sequence using LASTZ. Only the reads that aligned with at least 85% identity were selected and the difference between those reads and the assembled sequence was calculated. We used Velvet to generate the de novo assembly of the genome with a *k*-mer value of 21 and a coverage cutoff of 3.0. As can be seen in Table 2.2, YASRA generates an assembly that uses more reads, and generates contigs that align to reads with higher matches and lower mismatches than AMOScmp-shortReads. Solexa reads have more mismatch errors than indels (insertions/deletions), as is indicated by the low number of indel differences between the assembly and the reads, in case of all the assemblers. The utility of comparative assembly is again evident from the comparison of the assembled output from AMOScmp and YASRA when compared to Velvet, as they generate fewer, longer contigs, lower number of mismatches and indel errors. Velvet

Table 2.2: Comparison of the various assemblers for the assembly of finger millet plastid genome. We report the number of contigs (CN), the number of reads mapped, the number of nucleotide matches of the reads to the assembled sequence, the number of mismatches(MM), number of nucleotides extra in the assembled sequence and the number of nucleotides extra in the reads.

Assembler	CN	Reads	Matches	MM	Missing	Extra	Time(s)
Velvet	1,723	2,038,980	61,054,239	88,479	361	606	144
AMOScmp-shortReads	655	948,700	28,127,275	33,741	2527	354	2968
YASRA(yasra95short)	102	1,398,789	41,462,577	31,994	297	421	6285

on the other hands is able to use a higher fraction of the reads without the a-priori knowledge of the template genome. This dataset is a perfect example of how de novo and comparative assemblies should be treated as complementary approaches. This composite approach is explored further in Chapter 5.

The time taken by YASRA in this experiment is much more than taken by AMOScmp-shortReads and Velvet, but most of the time is spent in several iterations which are used to close gaps between adjacent contigs, and filter reads which could lead to multiple contigs for the same region of the reference genome (See Methods). A user switch allows selection of a single-iteration assembly for the finger millet which leads to formation of about 300 contigs in about 360 seconds, time comparable to that taken by Velvet.

2.2.3 Effectiveness of Assisted Assembly as the Template is Varied

An important parameter concerning the use of comparative assembly is the maximum evolutionary distance between the template and the target genome, that can be tolerated by the pipeline. We assembled the 454 reads for the non-HVR region of the mitochondrial genome of the woolly mammoth [30]. The reads corresponding to the M1 specimen were selected and assembled with four different templates and compared to the submitted mtDNA [Genbank:EU153444.1]. The closest template, which was at a little or no evolutionary distance from the target genome, was an earlier assembly of the mammoth mitochondrion [Genbank: NC_007596.2]. The second reference corresponded to the non-HVR region of the mtDNA of the elephant (*Loxodonta africana*) [Genbank:AJ224821.1], which is 95.10% identical to the target genome. The other two templates were the mtDNA of mastodon [Genbank:EF632344.1], which is 89.14% identical to the target genome and mtDNA of dugong [Genbank:AJ421723.1], which is 76.29% identical. The results are summarized in the Table 2.3.

We compared the performance of AMOScmp-shortReads, Newbler mapping assembler and YASRA on these templates. All the assemblers were able to assemble the correct sequence from

Table 2.3: The effect of variation of the template on the various assemblers. We analyze the number of contigs, the number of nucleotide matches to the correct sequence, the number of mismatches, the number of nucleotides missing in the assembled sequence and number of nucleotides extra in the assembled sequence.

Assembler	Reference	Contigs	Matches	Mismatches	Missing	Extra
Newbler	Mammoth	1	16,445	3	10	55
Newbler	Elephant	1	16,445	3	0	56
Newbler	Mastodon	24	12,902	2	4	42
Newbler	Dugong	9	1,708	0	0	0
AMOScmp-shortReads	Mammoth	2	16,444	0	4	0
AMOScmp-shortReads	Elephant	1	16,444	0	4	0
AMOScmp-shortReads	Mastodon	5	16,496	2	5	0
AMOScmp-shortReads	Dugong	18	4,566	2	1	0
YASRA(yasra95)	Mammoth	1	16,447	0	0	1
YASRA(yasra90)	Elephant	1	16,447	0	1	0
YASRA(yasra85)	Mastodon	1	16,447	0	1	6
YASRA(yasra75)	Dugong	1	16,447	0	1	0

the closest template but the performance of Newbler and AMOScmp-shortReads degraded as the templates were varied. YASRA on the other hand was able to assemble the correct sequence from all the templates with one of the default parameters.

2.3 Methods

Figure 2.1 outlines the steps taken by YASRA to assemble a genome, given a set of reads and a reference genome as input. We now discuss the various steps in greater detail.

2.3.1 Layout Generation

The first step in YASRA aims to determine the layout of the reads. This is achieved by aligning them to a template with LASTZ. We use low score thresholds to find weak matches to a distant template. Scores were tuned based on average aligned sequence identity and length, shown in Table 2.4. Tuning was achieved by intuition, and scores were tested on samples of about 10% of the reads, with a goal of balancing the number of reads for which a high-identity, high-coverage alignment could be identified vs computational time. Several datasets were used to tune and then test the LASTZ parameters for the different distances between the target and the template genome (See Table 2.4).

YASRA provides two choices for handling reads with more than one candidate placement.

Figure 2.1: Figure describing the YASRA pipeline.

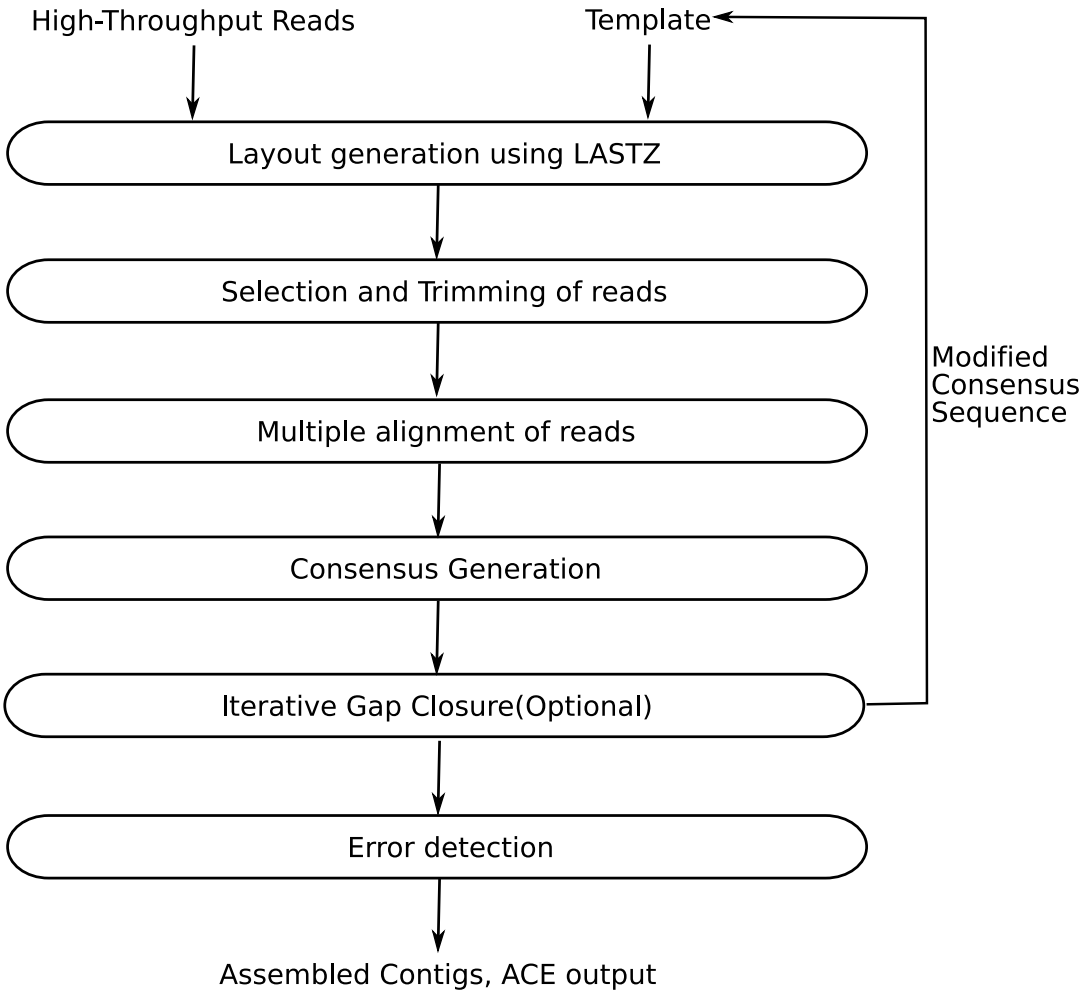


Table 2.4: Parameters of LASTZ based on percent identity and average read length. An experienced user can set different parameters based on other considerations. RL denotes the average read length and ID denotes the percent identity between the reference and the target.

ID	RL	LASTZ parameters	LASTZ switch
98	250	T=2 Z=20 --match=1,6 O=8 E=1 Y=20 K=22 L=30 --identity=98	yasra98
95	250	T=2 Z=20 --match=1,5 O=8 E=1 Y=20 K=22 L=30 --identity=95	yasra95
90	250	T=2 Z=20 --match=1,5 O=6 E=1 Y=20 K=22 L=30 --identity=90	yasra90
85	250	T=2 --match=1,2 O=4 E=1 Y=20 K=22 L=30 --identity=85	yasra85
75	250	T=2 --match=1,1 O=3 E=1 Y=20 K=22 L=30 --identity=75	yasra75
95	50	T=2 --match=1,7 O=6 E=1 Y=14 K=10 L=14 --identity=95	yasra95short
85	50	T=2 --match=1,3 O=4 E=1 Y=14 K=11 L=14 --identity=85	yasra85short

1. The user is allowed to set a threshold T. All the placements for a particular read, with scores in the interval $[max - T, max]$, where max is the score of the best placement (highest score), are marked and one of them is selected at random.

2. Reads with multiple candidate placements are not included in the assembly. If there are more than one equally high-scoring placements, then one of them is selected randomly.

Various other sources of information such as the mate-pair information for the reads, can be used in this stage to deduce the best placement for each read.

2.3.2 Selection and Trimming of Reads

The reads are trimmed at the ends of the last ungapped segment, on both ends of each read. The high coverage ensures that most of the low similarity regions between the target and the reference genome end up in the higher quality segment of at least a few of the reads. This allows us to use longer untrimmed reads for placement and in case of ancient DNA reads, to exclude damaged bases towards the ends of the reads. The trimmed reads are sorted in the order of their hit on the template, before they are output to the aligner.

2.3.3 Multiple Alignment of Reads

The reads are now supplied to an aligner in the order of their hit on the template. The algorithm used by the aligner can be enumerated in the following steps.

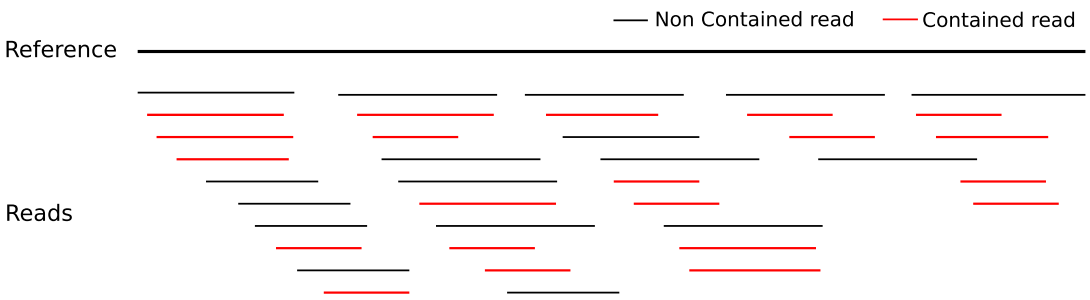
1. Separate the reads into “contained” and “non-contained” (Figure 2.2) reads; associating each contained read with a non-contained read. Typically more than 70% of the reads in a high-throughput assembly dataset are “contained” and separating them leads to a significant reduction in size and memory problems.

2. Use the set of non-contained reads to create an overlap graph with the reads as vertices in the graph. Align reads to other reads with an implicit overlap, and create an edge between two vertices if and only if their overlap score exceeds a predefined threshold.

3. Using the overlap scores as edge weights, find the maximum spanning tree of the overlap graph using Prim’s algorithm [48]. This gives a tiling path for the non-contained reads where each connected component in the graph describes the order of alignment of reads for a contig.

4. Create skeleton contigs, aligning the non-contained reads in the order as deduced in the above step.

Figure 2.2: Division of reads into contained and non-contained.



5. For each contig generated in the above manner, form a multiple sequence alignment by repeatedly aligning the associated contained reads in the order of their hit on the template.

The procedure described above gives a crude alignment, which is then processed by an iterative refinement procedure using a combination of two scoring schemes [49].

2.3.4 Consensus Generation

The refined multiple alignment is used to infer the consensus sequence based on a simple majority vote. This module can be replaced by another post-processor with a different definition of a consensus sequence. Other options include flagging of ambiguous base positions (heterozygous or poorly assembled), and generation of ACE files for easy viewing in Consed [50] or Hawkeye [51].

2.3.5 Iterative Gap Closure

One of the limitations of a comparative assembler are regions of high divergence between the template and the target genome. The default mode of assembly breaks the contigs when it encounters such regions, leading to a fragmented assembly. YASRA tries to close such small gaps, by “walking” on the contigs, and extending them, followed by attempts to merge adjacent contigs. This is an optional step, but we find that for smaller genomes, this helps in increasing the coverage of the contigs on the reference.

YASRA starts this step with the consensus sequence generated in the previous step. It creates a new reference sequence from the contigs, by joining them with a run of N’s. An approximation of the number of bases between any two contigs is based on their mapping on the actual reference sequence. Now YASRA iterates through all the steps in the pipeline starting from layout generation, to consensus generation, but using the new template as the reference. This process is terminated

if we get a single contig, or if a particular iteration leads to increase in the number of contigs or a decrease in terms of coverage on the original reference sequence.

2.3.6 Error Detection

This module is used as post-processor to the assembly process and breaks the contigs at regions of low coverage. It throws out contigs less than a predetermined length, as they might be spurious. It also looks for regions where the number of rejected reads by the assembler exceed the number of hits by more than a predefined threshold. Such a pile-up is indicative of an assembly error and this module breaks the contigs at those points. The result is an assembly which has more than a predefined coverage at all locations and is relatively free of assembly errors.

2.4 Discussion and Case Studies

YASRA was developed primarily for use with the Roche 454 family of sequencing instruments, for which the generated sequence reads are typically 200 bases or more in length. YASRA requires the availability of a reference sequence that is, say, at least 80-90% identical to the sequence being generated. The reads are aligned to the reference sequence, thereby determining overlaps among the reads, and then minor adjustments in the relative positions of the reads lead to a multi-sequence alignment and determination of the consensus sequence. It is important to note that large-scale rearrangements between the reference and the target genomes, can lead to a fragmented assembly; an issue which does not arise with de novo assembly. AMOScmp family of assemblers handle some of the large-scale rearrangements by identifying “fingerprints” of such events and then attempt to “undo” them.

YASRA has proved to be extremely useful, especially for assembling mitochondrial and chloroplast DNA. We used YASRA to assemble and report five complete mtDNA genomes of Siberian woolly mammoth [52]. Using a single mtDNA as the reference, YASRA was used to identify the differences in 18 mammoth mtDNA samples from the reference. The analysis of these differences was used to demonstrate the existence of two sympatric mtDNA clades for the mammoth. We used YASRA to assemble and report the first two complete mitochondrial genome sequences of the thylacine *Thylacinus cynocephalus* [53], a marsupial extinct since 1936. The assemblies were used to show the very low genetic diversity shortly before extinction of the species. We assembled the first complete mitochondrial genome sequences of the extinct ice-age woolly rhinoceros *Coelodonta antiquitatis*, and the threatened Javan *Rhinoceros sondaicus*, Sumatran *Dicerorhinus sumatrensis*, and black *Diceros bicornis* rhinoceroses [54]. A phylogenetic analysis of the 4 genomes together with the already published mitochondrial genomes of the white *Ceratotherium simum* and Indian *Rhinoceros unicornis* rhinoceroses, was presented. We assembled the mtDNA

of several polar and brown bears including mtDNA extracted from a pleistocene jawbone [55]. YASRA was also used to assemble mtDNA of several sloths (ancient and modern) and several Tasmanian Devil individuals [56]. YASRA is also being used by the project “Gymnosperms on the Tree of Life” [57] to assemble and analyze several chloroplast genomes.

Chapter 3

Alignment Clusters

Hierarchical assembly relies on reducing the problem of assembly of sequences, from a global problem (the entire genome) to a local problem (a single clone 40-200 kb in size). The genome sequence is then assembled by aligning sequences of adjacent clones and calculating a path through these alignments that will produce a non-redundant sequence. Evaluations of these alignments are guided by a tiling path [24].

A lot of assemblers that work with the next-generation sequencing technologies also work much better in context of a local problem. For example, Velvet reported contigs of significant length, upto 2-kb N50 for mammalian BAC's [40]. Similarly, Newbler can handle de novo assembly of genomes upto 1 Gbp in length [58]. Breaking the whole genome problem into a bunch of local problems helps in increasing the computational efficacy of the process, as the local problems can be analyzed separately. It can also help in separating out the most complex parts of the problem i.e. the repeat regions, from the initial analysis.

We discuss some of the approaches designed by the author to partition the whole genome problem into smaller, more-manageable problems. We define a "cluster", and then demonstrate simple techniques to form these clusters. The clusters are then assembled using software that can make the most judicious use of the several qualitative metrics offered by the sequencing technology.

3.1 Clusters

A "cluster" is a set of two linked items; a $(key, value)$ pair. The key, which is an unique identifier, is the header of a sequence referred to as the "reference", of the cluster. The $value$ consists of the following data:

1. The nucleotide sequence of the reference.

Figure 3.1: Example of a cluster data structure. The bases in bold depict the locations where the reads differ from each other. However only the first two locations are stored in the cluster, as they are the positions where the reads differ from the reference sequence. We store the reference base, alternate bases, position and number of reads supporting the alternate base for each difference.

```
Read0(Reference Sequence)    CTCTCTTCTTCTACGCGGTACTGTAAGT

Read1                              CTTCTTCTACGCGGTACTGTAAGTATGCTTAGCGGAT
Read2                                    TCTACGCGGTAGTGTAAGTATGCTAAGCGGAT
Read3                              CTTCTTCTACGCGGTAGTGTAAG-ATGCTTAGCGGAT
Read4                                          CGCGGTACTGTAAGTATGCTTAGCGGAT

Cluster:
reference header:    Read0
reference sequence: CTCTCTTCTTCTACGCGGTACTGTAAGT
parameters:          mismatches <= 2 bp, overlap >=15 bp, indels <=1
members:             Read1, Read2, Read3, Read4
differences:         20  C  G  2
                     27  T  -  1
```

2. The parameters/thresholds that are used in aligning the reference to other reads/contigs.

3. Header information of all the reads/contigs that align to the reference using those parameters/thresholds.

4. For each alignable read, the observed differences between the read and the reference sequence.

All this information can be gleaned from the “overlap graph” used in the traditional overlap-layout-consensus assemblers. An example of a cluster is shown in Figure 3.1. This data-structure can provide a local context to the assembly problem. Now we describe some of the methods where we use the clusters to infer locations of variation in a genome.

3.2 Clusters with a Reference

The Newbler mapping assembler [10] is part of the software suite that is supplied by Roche/454 with their sequencing instruments. When mapping the reads to the reference(s), it searches for suitable alignment(s) of the reads to the reference(s) in the nucleotide space. It then proceeds to construct contigs and a consensus sequence in flow-space. This consensus is then used to identify the putative differences between the reference and target sequence. It is designed to utilize all the quality metrics associated with the 454 generated reads and is under active development,

keeping in sync with the changes in the technology. However, it is extremely difficult to adjust the parameters for a new dataset. This problem is exacerbated as all the details of the algorithm used by Newbler are not known and it is in a state of constant flux.

One way to manipulate the output from Newbler is to use LASTZ [47] or some other aligner in the preliminary stage, to find the putative overlaps. Newbler can then be used to make fine adjustments to the multiple alignments, using the various quality metrics. The same technique can be employed for any other mapping software; however we describe the techniques used by the author for reads generated using 454 technology in the next section.

3.2.1 Methods

The reads generated using 454 are stored in a binary SFF format. The format is documented in the GS FLX documentation, page 445-448, which is supplied with the sequencing instruments. These SFF files serve as inputs to the pipeline. The steps taken by the pipeline to construct and assemble these clusters can be enumerated as follows:

1. The program “sffinfo”(part of the 454 software suite) is used to extract either all or a subset of the fasta sequences from a SFF file.

2. The extracted sequences are aligned to the reference sequence using parameters that are suitable for the expected rate of polymorphism between the reference sequence and the reads.

3. The reference sequence is divided into sub-sequences. This can be done, for example, by analyzing a self-alignment of the reference genome, to deduce intervals that can be uniquely mapped using reads of some length.

4. Each read is now assigned to one of the above calculated intervals.

5. We use a program “sfffile”(part of the 454 software suite) to create a single input SFF file for each interval, that only has the reads that were assigned to it.

6. The input SFF file is given to Newbler, which creates “micro-assemblies”. The generated ACE file is then read and the assembly is analyzed to find the locations of variation including SNPs and other large-scale operations between the reference and the reads.

3.2.2 Case Studies

We designed and implemented a pipeline, similar to the one described above, to call SNPs from a human individual [59]. The human reference (NCBI Build 36.1) sequence, henceforth referred to as

Table 3.1: Details of SNP calls from several human individuals. The calls were made using the pipeline described in Section 3.2. We report the number of found total differences and single base substitutions. We also report the number of novel substitutions found in comparison to dbSNP129. "gen" refers to fragment genome runs, and "mex" run refers to target resequencing runs to capture exomes.

Individual	Coverage	Num differences	Num substitutions	Novel substitutions
KB1	8.53X gen + 16X mex	3,882,615	3,453,440	973,712
NB1	2X gen + 16X mex	1,345,709	1,181,663	284,712
MD8	16X mex	140,261	125,848	35,294
TK1	16X mex	153,125	136,985	41,328

hg18 was divided into smaller intervals, determined by aligning hg18 to itself with high stringency and finding the intervals that did not align to some other part of hg18. The alignments were filtered to select the ones that were 300 bp or longer with at least 97% identity.

The fasta sequences were extracted from the SFF files and aligned to hg18. Alignment was performed using LASTZ with the following parameters:

```
--step=15 --seed=match13 --notransitions --exact=18 --maxwordcount=6
--match=1,3 --gap=1,3 --ydrop=10 --gappedthreshold=18 --identity=97
--coverage=90
```

The reads were allowed to align to repeats, by removing the soft-masking from hg18. A read with a single alignment, with identity $\geq$ 98%, was considered mappable. A read with more than one alignment, but with one with identity $\geq$ 98% and at least 1.5% better than the second best, was also considered mappable. Each interval was assigned a designated reference, which was the bases in hg18 which belonged to the interval with 500 bp appended on each end. The reads corresponding to an interval were collected and the information about them extracted from the original sff files. This included the complete untrimmed sequence, the quality values and the flowgram information. This information was then combined to form an input sff file, and supplied to Newbler mapping assembler as described in the Section 3.2.1. The SNP calls made by Newbler were then transferred to original hg18 coordinates and then filtered to remove any regions which could have been part of a repeat, leading to a pile-up of reads.

Table 3.1 describes the details of the SNP calls. The KB1 sample was also genotyped using Illumina Human 1M-Duo BeadChip [60]. We were able to verify the calls on 1,108,040 positions of the genome and calculate the error rates for our pipeline. The reported false-positive rates are pretty low around 0.07% for the sample, in comparison to 0.38% for the Watson genome [61] (calculated based on number of mismatches with dbSNP).

3.3 Clusters without a Reference

Roche/454 also supply a de novo Newbler assembler with their sequencing instruments. It calculates the overlaps between reads, creating contigs by constructing multiple alignments of the reads that tile together. It then generates the consensus base-calls by averaging the processed flow-signals for each nucleotide in the alignment. Despite being an excellent tool, it suffers from the same drawbacks as the Newbler mapping assembler. The read lengths of sequences generated using 454 technology allow us to use algorithms and data-structures similar to that in Section 3.2, to find the putative overlaps and deduce the variations in the genome being sequenced.

3.3.1 Methods

The steps taken to find the locations of variation can be enumerated as follows:

1. The program “sffinfo” is used to extract either all or a subset of the fasta sequences from the SFF files.

2. The reads are aligned to each other using LASTZ with the following parameters:

   ```
   W=13 Z=26 --exact=40 --match=1,3 --gap=1,3 --ambiguousn --format=maf
   --identity=96 Y=10
   ```

   These parameters were tuned for reads with an average length of about 300 bp. Tuning was achieved by intuition, and the parameters were tested on small sub-samples of datasets.

3. We create a “cluster” (Section 3.1) corresponding to each extracted read in the dataset.

4. Expected coverage values can be used to filter these clusters, either to keep or remove clusters that exceed certain pre-computed thresholds.

5. The program “sfffile” is then used to create an input SFF file for each cluster.

6. The SFF file is fed to Newbler de novo assembler, which creates “micro-assemblies”.

7. The resultant contigs are blasted against databases to determine if they are contaminants or if they belong to the species being sequenced. They can also be analyzed for heterozygous locations to find the locations of variation in the sequenced genome. This concept is further explored and extended in Chapter 4.

3.3.2 Case Studies

We used the steps described in Section 3.2 to implement a pipeline to call SNPs from the genome of a human individual. The reads that did not align to hg18 were then assembled de novo, using the strategy mentioned above. For KB1, 7,111,129 of the 79,637,936 reads did not align to hg18 using the parameters described in Section 3.2.2. A lot of these reads are expected to be from regions that have not yet been successfully assembled in hg18. Some of them could be contaminants while other could belong to interesting regions of the genome where KB1 differs considerably from the human reference. These reads were aligned to each other and we generated "clusters" using the overlap information. We filtered to keep the clusters with 4-35 reads, for an expected coverage of around 8.5X. This left 14,079 clusters comprising of 268,888 reads. The resulting assembly of these clusters using Newbler gave 15,052 contigs spanning 8,330,341 bases.

The sampled individuals are the indigenous people of southern Africa, belonging to one of the 14 "ancestral population clusters" [62]. The expectation was that we would find some contigs that align more closely with the chimp genome, than to the human reference sequence. We aligned these contigs to the human reference assembly, hg18 and the chimp assembly (Build 2 Version 1, Oct. 2005) using the following LASTZ parameters:

```
--match=1,20 --gap=21,20 Y=221 L=102 --ambiguousn --identity=97
--coverage=80
```

The alignments were collected and another program was written to find the contigs that aligned to chimp but had no alignments to hg18 with the given parameters. We found 1,840 such contigs, most of which aligned to hg18 at a lower identity and coverage threshold than to chimp. Further analysis has revealed some interesting locations and variations in the genome and the results shall be discussed in another publication [59].

Chapter 4

DIAL

Resequencing has been the driving force in the acceptance and growth of the next generation of sequencing instruments. Hence, it does not come as a surprise that most of the algorithmic advances have been made in read-mapping and identification of polymorphism using those mappings. Much less has been published about identifying genomic differences in species where we lack an assembled reference sequence. This chapter discusses the design and utility of DIAL (De Novo Identification of Alleles), a program written by the author to deduce genetic differences within the target species without the help of a reference genome. The approach works even when the depth of coverage is insufficient for de novo assembly. DIAL's performance on several extensive datasets is described and evaluated.

The design of the software was motivated by the problem to identify a sufficient number of single-base differences to design a custom genotyping array using only as much sequence data as necessary. With current array technology, at most a few thousand differences can be assayed at an affordable cost (a few tens of thousands of dollars), so our main interest is in knowing how to identify that many differences with a low false-positive rate, and in understanding the main sources of error at low levels of sequence coverage.

4.1 Results

We give a brief description of the computational problem and our approach to handle them. We evaluate the software's effectiveness using published Roche/454 sequence data from the genome of Dr. James Watson (to detect heterozygous positions) and recent Illumina data from orangutan, in each case comparing our results to those from computational analysis that uses a reference genome assembly. We also illustrate the use of DIAL to identify nucleotide differences among transcriptome sequences.

4.1.1 Overview of the Computational Problem

Our strategy will be to observe the fragments of DNA sequence data produced by a sequencing machine and identify pairs of fragments that overlap, but differ in one or several positions. When the number of identified variant positions reaches a desired level (depending on the application at hand), sequencing can stop. If the fragments are all from the genome of one individual, then we are looking for heterozygous positions (i.e., different nucleotides from each parent), whereas with sequence fragments from more than one individual, we can also find positions where every individual is homozygous (identical maternal and paternal nucleotides), but at least two individuals differ. The differences we seek are typically single-nucleotide substitutions that occur within a species or population of interest. We call each position at which different nucleotides appear among the individuals being sequenced a "SNP", short for Single Nucleotide Polymorphism. Each variant nucleotide that appears at a SNP we call an "allele"; for this paper we make no requirements concerning the frequency of an allele in the population, other than that it occurs in our samples, and we assume there are precisely two alleles at each SNP. Our approach can also detect insertions/deletions of one or several nucleotides, but these are of less interest to us because high-throughput genotyping methods (i.e., techniques to efficiently assess which variants are possessed by each of many individuals) focus on nucleotide substitutions. Here we describe two difficulties with our strategy, explain how we handle them, and analyze their effects on our results.

The first difficulty is dealing with sequencing errors, which can masquerade as SNPs. The actual frequency of SNPs depends on the genetic diversity of the species being sequenced. For instance, the frequency of SNPs between two humans averages around one per 1,000 positions, but SNPs in some species are much less frequent. In any case, the rate of errors in individual fragments produced by the sequencing instrument (regardless of the brand) is probably much higher than the rate of SNPs. This means that when we see a nucleotide difference between two overlapping reads, odds are that it is a sequencing error rather than a true SNP. A cornerstone of our approach is to require that each allele of a putative SNP be observed at least twice, which we believe is significantly more likely to indicate an actual difference than a repeated sequencing error. (Think of sequencing errors as being approximately independent and having a 1% frequency, so a repeated error will strike at 1 in 10,000 positions, compared to, say, 1 SNP per 1,000 positions.) This requires the genomic position to appear in at least two fragments with one allele and two fragments with the other, and raises the following question: Given X-fold genome coverage in random fragments of an individual genome (i.e., the fragments' total length is X times the size of the genome), what fraction of the genome positions will be contained in two or more fragments from each parent? Or similarly when looking for homozygous differences between two individuals that are each sequenced to depth X/2? In theory, the answer is z^2, where $z = 1 - e^{-X/2}(1 + X/2)$

[63]. For instance, given 1-fold coverage of an individual, the expected genome fraction where heterozygous positions can be identified is 0.0081. Thus, if a genome has 1 million heterozygous positions, we expect around 8100 of them to be detectable with 1X sequence coverage if no other issues are considered. Under the same conditions, 2X coverage should put 70,000 SNPs within reach of our method. For a genome with fewer heterozygous positions, our potential yield is proportionately less, e.g., 810 SNPs from 1X coverage of a genome with 100,000 heterozygous positions. Typically, we will not know the SNP density in the sequenced individuals when we start a project; we are more likely to know the number of SNPs that can be accommodated on a genotyping array that is within our budget.

The second difficulty is that overlapping reads differing at one position may not actually be from the same part of the genome; instead they may be from the two copies of a recent duplication that have accumulated a mutational difference. In that case, a detected difference is of no use for a population-genetic analysis because every individual genome of that species may contain both variants and thus appear to be heterozygous at a single location in a genotyping assay. Hence we consider a SNP prediction to be an error if it turns out to be caused by our failure to distinguish copies of a repetitive region. Our main defense against this possibility is to limit the number of sequenced fragments that contain a given SNP. If a genomic interval has hundreds of nearly identical copies throughout the genome (which can easily happen with families of interspersed repeat elements, such as Alus in primate genomes), even with only 0.5X coverage we will see a pile-up of overlapping reads. The problem comes from regions that have only a few nearly identical copies in the genome. To give a sense of the frequency of low-copy repeats in one particular species, consider NCBI Build 36 of the human reference genome. There, 2.16% of the bases are in a repeated region (defined here as at least 97% identity over 300 bases) with precisely 2 copies; for 3 or 4 copies the fractions are 0.72% and 0.43%. It is not obvious how to extrapolate that estimation to the parts of the human genome not represented in the reference assembly (but which may be represented in next-generation sequenced data), or to another species. However, success of our method clearly requires that the fraction of the genome contained in high-identity, low-copy-number repeated regions be very small, particularly since the fraction of nucleotides in a duplicated region that differ among the copies is likely to exceed the fraction of positions in single-copy regions that are SNPs.

These two difficulties lead to competing constraints on our strategy; we need to thread our way between not having enough overlapping reads and having too many. Moreover, this threading must be done in the dark, because we typically will not know all of the critical parameters, including the number of actual SNPs among the individuals being sequenced and the frequency of recent genomic repeats in the target species. However, we can obtain guidance by using theoretical models that we fit to observed data. For instance, one decision that needs to be made is the upper limit,

Table 4.1: Numbers of heterozygous positions correctly identified, and numbers of false-positive predictions from low-copy-number repeats expressed as a percentage of the identified SNPs. The values are given as a function of fold coverage (X, row labels) and the upper bound on the number of overlapping reads (x, column labels). For instance, at X = 1.0 and $x = 5$, there are 6,131 correct SNP calls and 37.9% as many duplication-induced erroneous ones. This is a theoretical analysis based on informally fitting a model (see text) to data from the genome of Dr. James Watson.

X\x	4	5	6	7	8
0.5	473/54.8%	552/65.0%	561/68.3%	561/69.0%	561/69.1%
1.0	4,598/28.6%	6,131/37.9%	6,450/43.5%	6,501/45.9%	6,508/46.7%
1.5	14,119/14.9%	21,179/21.4%	23,386/26.8%	23,915/30.3%	24,021/32.0%
2.0	27,067/7.8%	45,111/11.8%	52,630/16.0%	55,036/19.5%	55,675/21.8%
2.5	40,080/4.1%	73,481/6.5%	90,877/9.4%	97,835/12.2%	100,145/14.6%

call it x, on the number of reads that appear to contain a SNP. For identifying heterozygous SNPs in a single genome, the fraction that can in theory be identified when requiring two observations of each allele and at most x overlapping reads, given X-fold coverage of the genome, is:

$$P(X) = e^{-X} \sum_{j=4}^{x} X^j (1 + (j+1)2^{1-j})/(j!)$$

For intervals that appear precisely twice in the genome (albeit with a small number of differences), the fraction of differences that will be mis-identified as SNPs is approximately $P(2X)$ [63]. If we assume there are 800,000 true SNPs in single-copy regions, 30,000 differences in duplicated regions, and 2.3 copies of a duplicated region on average, then Table 4.1 shows the numbers of correct positions and the relative number of copy-induced false positives; these numbers roughly reflect data observed for humans.

This model shows the following general trends. The main trend in the percentages is a difference among the rows. Namely, at 0.5X coverage, the number of incorrect calls caused by low-copy repeats is roughly 60% of the number of correct calls, but as the coverage increases, that percentage drops significantly, reaching around 10% at 2.5X coverage. A less dramatic but still noteworthy trend is seen within rows: at 0.5X coverage, allowing more than $x = 5$ overlapping reads makes little difference, while at 2.5X coverage, setting $x = 6$ produces substantially more correct SNP calls, but with a substantially higher false-positive rate. (For even higher levels of coverage, higher values of x are appropriate, and DIAL adjusts automatically; see the Methods.) The specific values in the table depend heavily on the assumed numbers of SNPs and of differences within low-copy repeats, but similar trends are observed over a wide range of conditions and provide insight into the problem at hand.

Our goal is to perform only as much sequencing as is needed to produce a desired number of SNP predictions with an acceptable false-positive rate. In practice, we impose additional restrictions on our SNP predictions, as described in the Methods section, but the two main conditions are that we see each putative allele of a SNP at least twice, and that there not be too many reads that appear to overlap the SNP. The additional conditions, such as high quality scores and lack of other SNP predictions in the assembled region around the SNP, are intended to decrease the fraction of erroneous SNP calls and to provide predictions that are appropriate as input for certain experimental protocols (such as PCR amplification of the position). However, the extra restrictions unavoidably reduce the number of correct SNP calls.

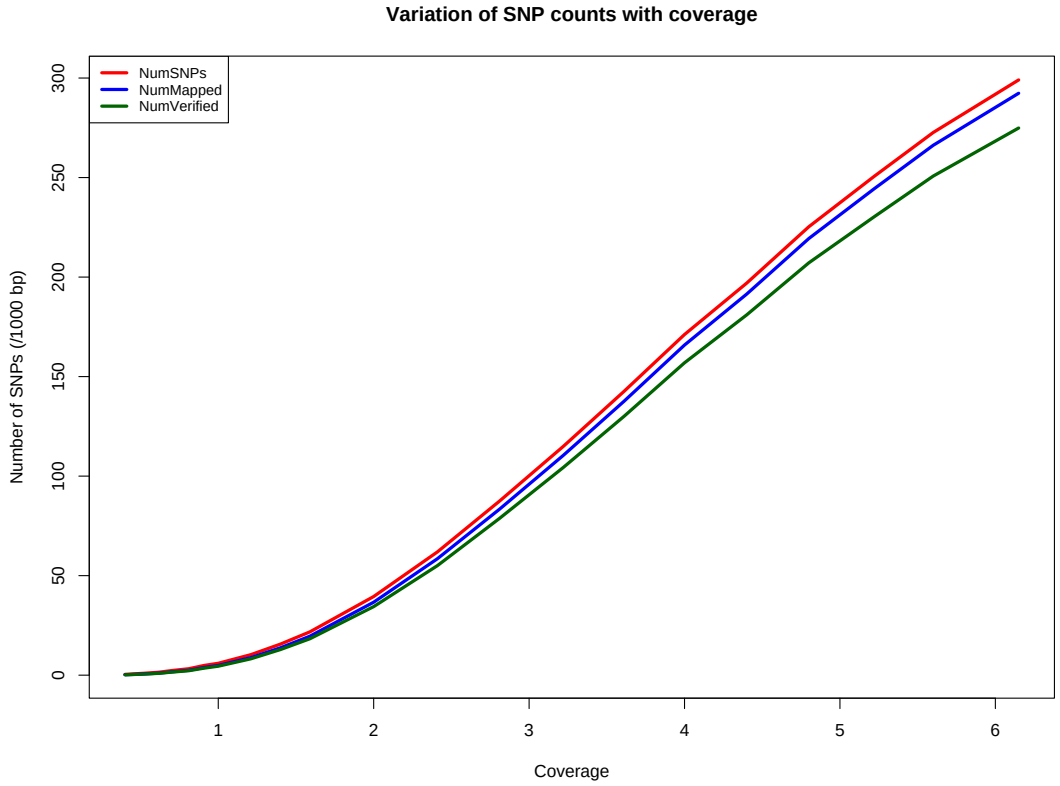
4.1.2 Watson Genome Dataset

The genome of Dr. James Watson was the first full genome to be sequenced using a next-generation sequencing technology [61]. Several groups predicted single-nucleotide differences from the human reference sequence, including the predictions of heterozygous positions that we used to evaluate the accuracy of our SNP-calling method. Fortunately, we have an extremely accurate assembly of a human reference genome sequence, making it possible to reliably determine which predictions lie in a near-identical, low-copy-number repeat, which are the bane of our strategy for calling SNPs without a reference (at least at low levels of sequence coverage).

We proceeded as follows. Of the 106.5 million high-quality reads that were generated for the Watson project, 76.6 million are available in the Short Read Archive at NCBI. The available reads total 18,952,140,632 bases and provide approximately 6.15-fold coverage of the genome. We downloaded the corresponding SFF files (low-level representations of the sequence data produced by Roche/454 sequencing machines, which include information used by our DIAL pipeline to improve SNP-calling accuracy). DIAL processed this data in about 350 CPU hours on a workstation. Starting at 0.4X coverage, and then at regular increments, DIAL predicted heterozygous positions. Along with each SNP prediction, DIAL reports an interval of assembled sequence that contains the putative variant position. Our subsequent analysis focused on high-confidence SNP calls where no other nearby SNPs were predicted (details in the Methods section); this was done because our main intended application for DIAL is to make predictions used for subsequent experimental studies, which frequently require variant-free sequences on both sides, e.g., for PCR primers. However, it also helps to eliminate low-copy-number duplicated regions that split more than a few million years ago, because older duplicates will probably have accumulated multiple nucleotide substitutions. (Though in some cases, subsequent gene-conversion events may have erased differences from an older duplication.)

Figures 4.1 and 4.2 show the number of heterozygous positions that DIAL predicted for the Watson genome at various levels of sequence coverage. For instance, at 1X coverage DIAL iden-

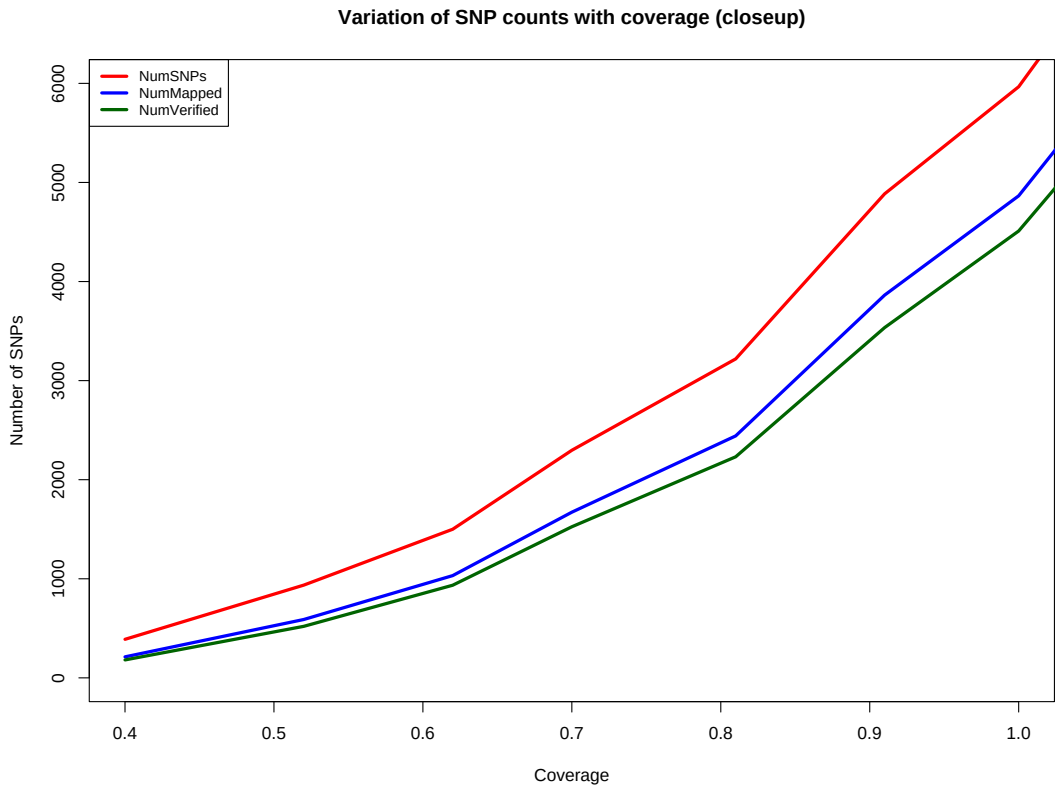
Figure 4.1: A plot showing the variation of several results as a function of sequence coverage. “NumSNPs” is the number of SNPs called by DIAL. “NumMapped” is the number of SNPs and their assembled flanking regions that could be uniquely mapped with greater than 98% identity to NCBI Build 36 of the human genome. “NumVerified” is the number of DIAL SNP calls that were reported for the Watson genome by Cold Spring Harbor Lab, ENSEMBL, or dbSNP.



tifies 5,966 potential SNPs, while at 2X and 6.15X it identifies 39,502 and 299,064, respectively.

We judged that a heterozygous SNP predicted by our method is correct if (1) the accompanying assembly of flanking sequence aligned to precisely one position in the human reference genome (NCBI Build 36) at appropriately high thresholds (98% identity over 200 bp) and (2) it agreed with any of the Watson heterozygosity calls that we downloaded from three sources (Cold Spring Harbor Laboratory, ENSEMBL, and dbSNP). Figures 4.1 and 4.2 plot the number of SNP calls, the number of SNP regions that appeared to be unique (criterion 1), and the number of these that were verified against the heterozygous positions found by other groups using a reference-mapping approach (criterion 2). Figure 4.1 indicates that for high genome coverage, the rate of erroneous SNP

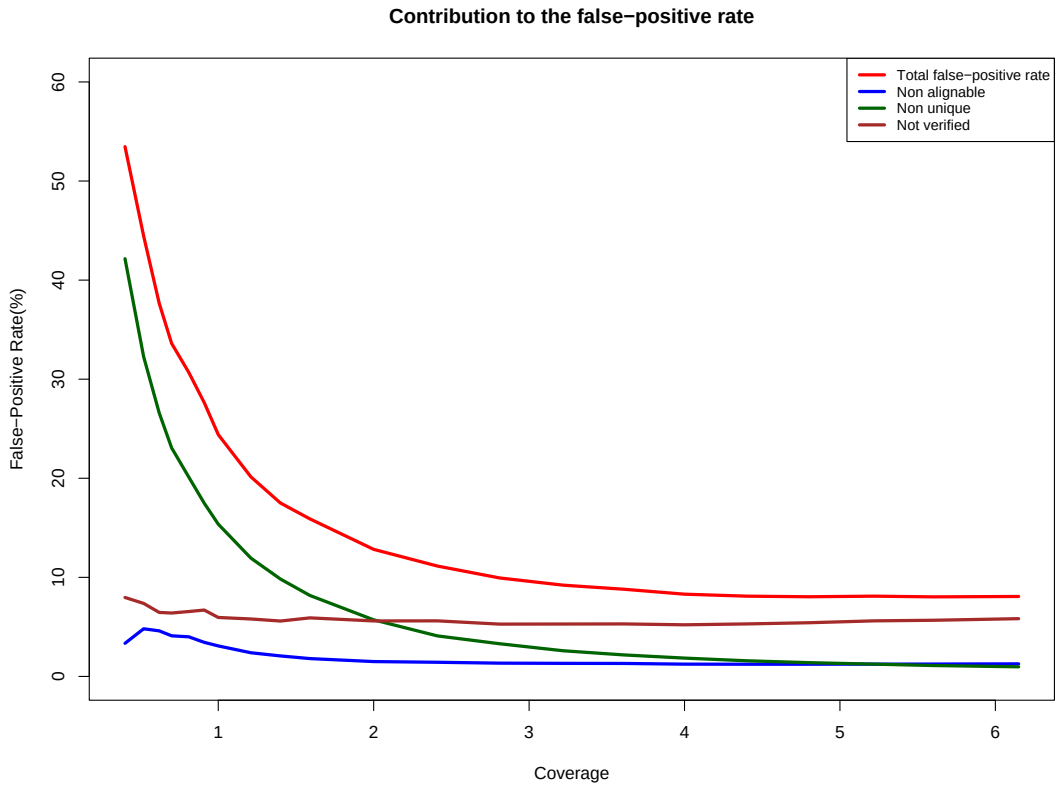
Figure 4.2: Magnification of the data in Figure 4.1 for 1-fold coverage or less.



predictions is around 10%, over half of which appears to be over-prediction in single-copy genomic regions (NumMapped – NumVerified in Figure 4.1). However, of the 17,300 SNP predictions that DIAL made in single-copy genomic regions at 6.15X coverage but which were not in the other sets of Watson SNP calls, 43% were found in other databases of SNPs, which suggests to us that many and perhaps most of the putative false positives are actually incomplete heterozygosity calls in the other Watson SNP collections. Figure 4.3 shows the various sources of error in the false-positive rate. At 0.5X coverage of the Watson genome, over 50% of DIAL's heterozygosity calls were judged incorrect; the main source is low-copy-number genomic repeats, as discussed above. The coverage and difference filters applied to mark clusters of reads as "candidate repeats" work much better as the coverage increases. A small fraction of the assembled neighborhoods of putative SNPs fail to align to the human reference genome at our requirement of at least 98% identity, though in general they align under relaxed conditions.

We also considered the number of false negatives, i.e., heterozygous positions in the Watson

Figure 4.3: “Not verified” means that the SNP was not found in the databases of Watson’s heterozygous positions that we consulted. “Not alignable” means that the assembled neighborhood of the SNP did not align to the human reference at 98% identity or higher over at least 200 bp, whereas “Not unique” means that it aligned more than once.



genome that seem to meet the conditions we impose but are not identified by DIAL. We estimate that there are about 1 million heterozygous positions that do not lie in a repetitive region or near another heterozygous position, but only 299,064 are found by DIAL from 6.15X of sequence data. The reasons for this discrepancy include the difficulty of predicting heterozygous positions with only low coverage data and the variety of additional conditions (see Methods) that we require before calling a SNP.

4.1.3 Orangutan Dataset

Illumina sequencing technology generates short reads only 75 bp long, but the amount of data generated in a single run far exceeds the data generated in a run by the 454. We used a single

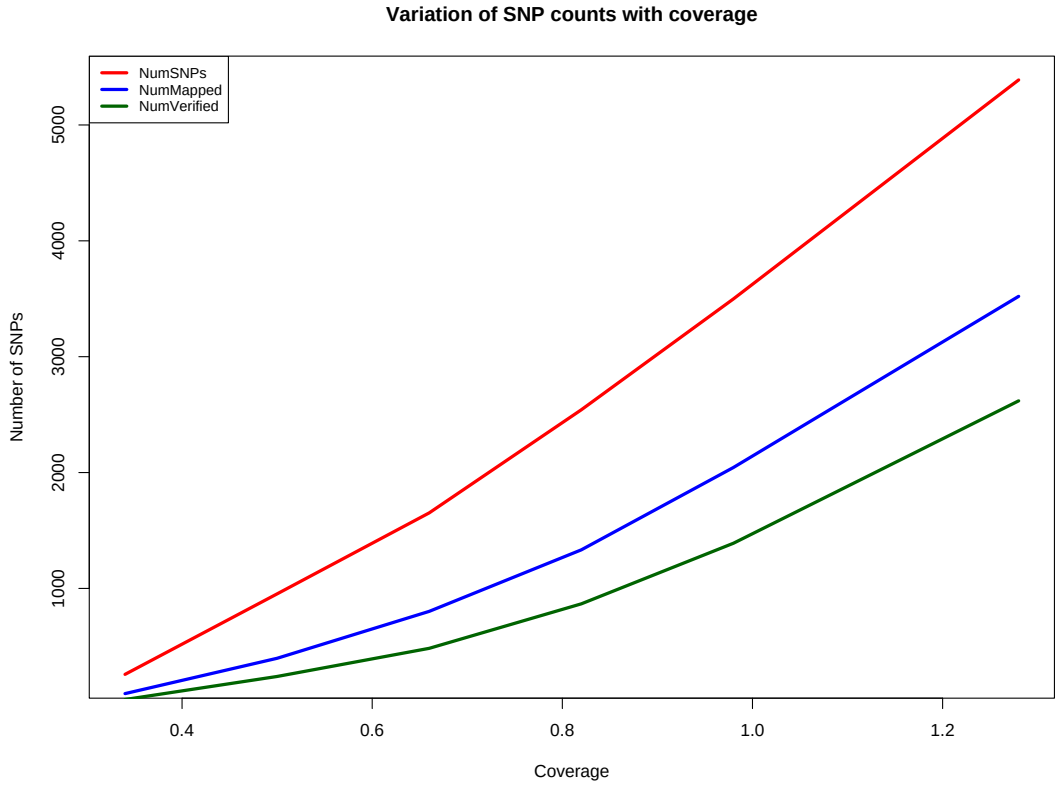
run of 75 bp reads from a Bornean orangutan *Pongo pygmaeus* to call heterozygous sites and evaluate our approach on Illumina reads. We adjusted the alignment parameters in the pipeline to make them more amenable to the short reads. We were able to identify 5389 high confidence SNPs for this dataset. A consequence of the short read lengths is that the assembled contigs for the orangutan reads are about 90 bp, whereas the flanking regions for the Watson genome dataset were around 350 bp. As for the Watson dataset, we plot the number of SNPs called by DIAL and the SNP locations which could be verified in Figure 4.4.

We had SNP calls from various outside sources for the Watson genome dataset; however no such calls exist for this one. Hence, we verified our calls against a mapping of reads on the Orangutan draft assembly (WUSTL version Pongo_albelii-2.0), which we made using MAQ [25]. An allele was deemed to be correct if it was supported by at least one read on the assembly with a mapping quality greater than 0. Even though we make a comparable number of SNP calls for the Orangutan and the Watson dataset, the false-positive rate for these shorter reads is much higher, as deduced by the comparison to the results from MAQ. In order to investigate the reasons for this difference, we mapped a single run of 75 bp fragment reads from the genome of a Korean individual [64] to the human assembly. MAQ was able to align 91.9% of the reads, mapping 86% of them uniquely on the hg18 assembly. In contrast to that, MAQ was able to align 89% of the short reads, but mapping only 75.9% of them uniquely on the Orangutan draft assembly, much lower than that for the human assembly. As shown in Figure 4.4, the biggest source of false-positive rates for this dataset is the number of assembled contigs that could not be uniquely mapped. Most of the contigs which could not be uniquely mapped map to the random chromosomes. We believe that as the Orangutan assembly improves, the bases in the random chromosomes will get assigned and annotated. That should substantially lower the false-positive rate for this dataset.

4.1.4 Mule Deer Transcriptome Dataset

Compared to generating sequence data from an entire genome, sequencing transcribed DNA (cDNA derived from messenger RNA) has several potential advantages. For a typical mammalian genome, the total length of all gene transcripts is approximately 1% of the genome's length; for mammals this means roughly 30 million bases instead of 3 billion. Thus, much less sequence data is required to reach depth-4 coverage for a substantial fraction of the target. And for some applications, the much higher fraction of functional sites is useful. However, there are disadvantages as well. While blood is a typical source of DNA for whole-genome sequencing, the number of genes transcribed in blood cells is rather low. If possible, it may be preferable to sequence DNA from another tissue or combination of tissues. Another complication is that the relative amounts of transcript DNA vary widely among genes, which lowers the efficiency of transcript sequencing (because of redundant data from highly expressed genes), and forces us to remove our constraint

Figure 4.4: A plot showing the variation of several results as a function of sequence coverage. “NumSNPs” is the number of SNPs called by DIAL. “NumMapped” is the number of SNPs and their assembled flanking regions that could be uniquely mapped with greater than 96% identity to the Pongo_albelii-2.0 assembly of the orangutan genome. “NumVerified” is the number of DIAL SNP calls that were verified as present in the read mappings.



on the maximum number of fragments that overlap a SNP.

To experiment with this approach, we applied DIAL to 192 Mbp of Roche/454 transcript data obtained from a pooled (i.e., from several individuals) lymph-node cDNA library from mule deer (*Odocoileus hemionus*). Despite the reduced number of genes transcribed in lymph nodes compared to certain other tissues (e.g., brain, testis), we identified 415 high-confidence SNPs, which is sufficient to populate a small genotyping array with SNPs having a good chance of relating to differences in immune-system genes with phenotypic effects. Note that a similar amount of whole-genome DNA would provide only about 0.06-fold coverage, which according to our results discussed above is expected to reveal less than 1 SNP, assuming a total of about 1 million nu-

cleotide differences in the sampled mule deer.

4.2 Methods

Our pipeline proceeds in several steps. Sequences from one or more individuals, and from one or more runs, serve as input to the pipeline. The approach is summarized in Figure 4.5 and described in greater detail below.

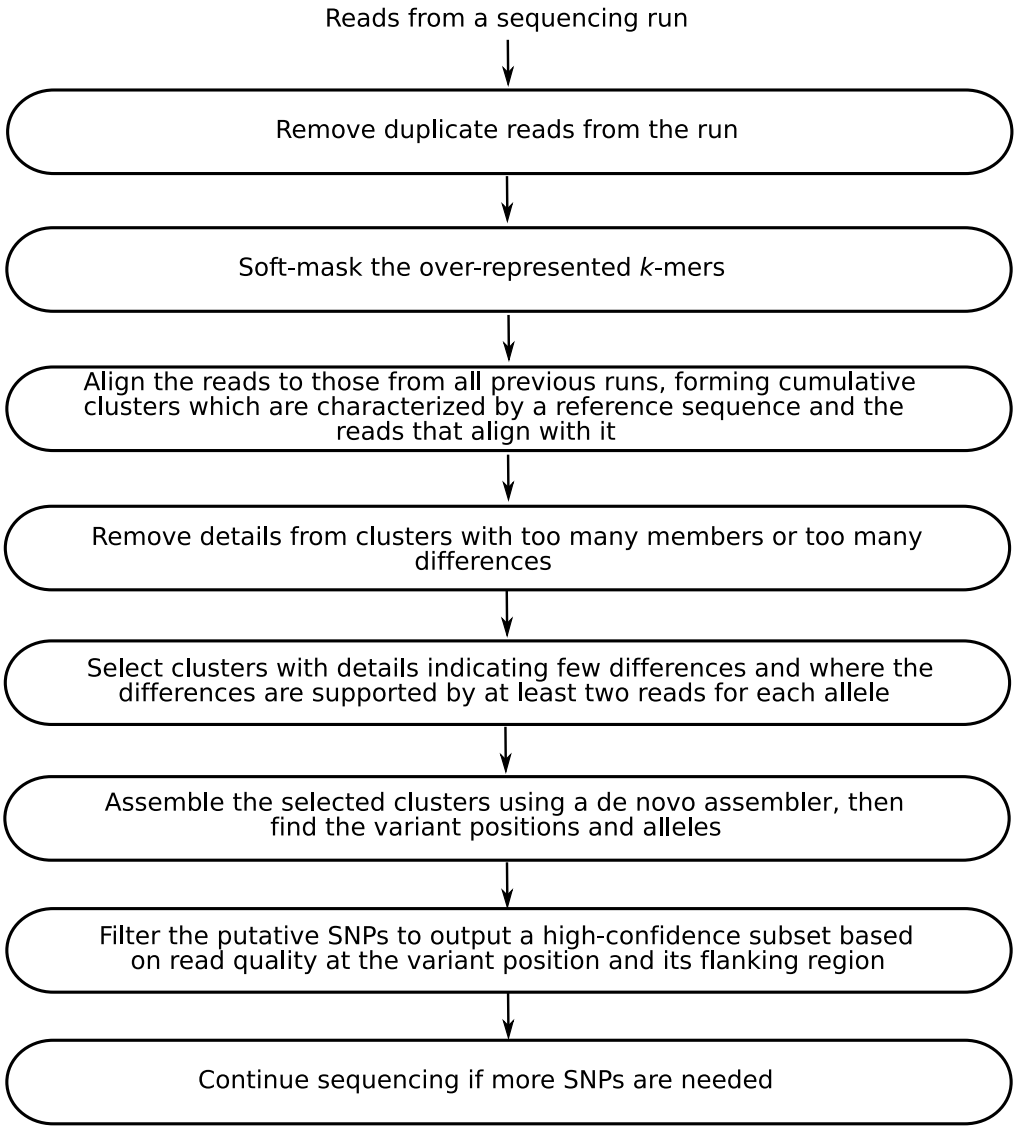
4.2.1 Masking of Repetitive Sequences

Duplicate reads of a single segment of DNA are a known potential artifact of emPCR (which is used by Roche/454 and Illumina/Solexa) and can lead to multiple identical reads with the same outer coordinates. This can happen, e.g., when two or more beads are amplified in a single emulsion microreactor. These "clones" can lead to erroneous SNP calls and coverage assessments. The first step in the pipeline is to collapse all of the "clones" of a read into a single sequence, which we accomplish by use of a simple hash function. We identify reads with the same md5 checksums [65] and if they have identical bases, then we replace them with a single sequence.

We then feed the non-duplicate reads to another module, which is nearly identical to WindowMasker [66]. WindowMasker uses linear-time scans of a genome to identify repetitive sequences, and has been proven effective in cases where much of the sequence is in draft form. We modified the algorithm so it could be used in conjunction with reads from sequencing runs. We soft-mask probable repeats (i.e., over-represented k-mers) by using lower-case letters for nucleotides, so that alignment anchors are not chosen in those regions. This process consists of three sequential passes.

1. The first pass calculates appropriate parameters and cutoff values. We define K, the size of the k-mers to be used, as the largest integer such that $(N/4^K) > 5$, where N is the sum of the lengths of the input sequences from that run. We calculate and store the frequency of each k-mer, $freq(S)$, in a splayhash for efficient access, then sort the k-mer frequencies and store the results in a temporary array F. If C is the total number of unique k-mers from the sequences in the run, then cutoffs T_{high}, $T_{threshold}$,T_{extend} and T_{low} are calculated as $F[int(0.998 * C)] + 1$, $F[int(0.995 * C)] + 1$, $F[int(0.99 * C)] + 1$ and $F[int(0.9 * C)] + 1$, respectively. In order to mitigate the effects of a few k-mers with very high frequencies and to avoid keeping track of counts for a lot of infrequently occurring k-mers, we compute a

Figure 4.5: Figure illustrating the DIAL pipeline.



score for each k-mer, defined as:

$$Score(S) = \begin{cases} T_{high} & \text{if } freq(S) \geq T_{high} \\ \lceil T_{low}/2 \rceil & \text{if } freq(S) < T_{low} \\ freq(S) & \text{otherwise} \end{cases}$$

2. The second linear scan divides each sequence into a set of overlapping windows. Each

window contains 5 k-mers, and the score of a window is defined as the average score of the k-mers in it. Adjacent windows share 4 k-mers and we mask the contents of a window if it has a score greater than $T_{threshold}$. We also mask the intervals between two masked windows, if the score of all the windows in it exceeds T_{extend}.

3. The last pass of the masker looks for local repeats. We calculate the frequency distribution of the k-mers within a read, and if any of the frequencies exceed T_{extend}, then we mask the whole read. This is particularly effective in reducing the time required to calculate overlaps between read sequences. This pass also unmasks any masked segment less than 40 bp long; the motive here is to unmask the high-frequency k-mers in non-repeat sequences.

4.2.2 Calculation of Overlaps

We align the masked sequences from the input run against all of the masked reads from the prior runs, using LASTZ [47], which uses a "seed and extend" strategy so that most of the spurious comparisons are discarded at the outset. The appropriate scoring matrix for LASTZ varies depending on the error model of the sequencing technology used. Here we report the details of the score-set we use for sequences generated with the 454 technology [10]. We set the match score to 1 and the mismatch score to 3, along with gap open and gap extend penalties of 1 and 3 respectively. This is in concordance with the error model for 454 reads, where indel errors are more frequent than substitution errors. We require a perfect overlap of at least 40 bp between two reads for them to be considered for gapped extension. We find the endpoints of the alignment by terminating the gapped extension whenever its score falls 10 points below the maximum score observed so far for that alignment. Two reads are called "aligned" if they have an overlapping segment of at least 100 bp and an identity of 96% or more in the overlapping segment.

We modified this score-set for use with the shorter Illumina reads; the gap open and gap extend penalties are both set to 10, while the match score is 1 and the mismatch score is 10. We find the endpoints of the alignment when the score falls 20 below the maximum score observed on that path. Two reads are termed as aligned if they have an overlapping segment of 50 bp and an identity of 96% or greater.

Every sequence in the input has a corresponding "cluster" where we store its header and sequence along with the names and observed differences of all the other reads that align to it. These clusters are cumulative over the sequencing runs. We mark the clusters as "candidate repeats" if they exceed thresholds, both in terms of the number of members (aligning reads) and the number of observed differences. This is done to prevent a combinatorial explosion that would occur if we stored all the details of all the clusters. We stop adding reads to the cluster once it is marked as a candidate repeat, even though the reference read of a cluster can still be added

to other clusters if it aligns with their sequences. The only exception to this filter is in the case of transcriptome datasets, as transcripts have variable abundance depending on their expression in the sample.

If the average depth-coverage is C, the coverage k at each base of the genome approximately follows the Poisson distribution

$$f(k|C) = \frac{C^k e^{-C}}{k!}$$

DIAL calculates a value $C_{threshold}$ that equals the 99.5^{th} percentile of the cumulative distribution function for the coverage depth. When a run is added and the number of members in a cluster exceeds $C_{threshold}$, then the cluster is marked as a candidate repeat. A cluster is also marked as a candidate repeat if the number of differences in it exceeds a limit $D_{threshold}$. For the reported datasets, this threshold was set to 10 bp.

4.2.3 Determination of SNPs

We can call SNPs after the overlaps for a single input run are calculated. The determination takes place in several steps:

1. First, we filter the clusters to consider only those which have substitution differences supported by at least two reads for each allele. Such clusters with more than 3 substitution differences, or with two or more such differences that are within 50 bp of each other, are discarded. If the input to the pipeline consisted of sequences from more than one individual, then we also filter to keep only clusters that have at least two reads from each of at least two individuals. This step also discards clusters which were marked as “candidate repeats”; they are not considered in the SNP calling stage.

2. We then iterate through the filtered clusters and select those with members which have not been observed in any of the clusters selected prior to it. This ensures that a read appears in at most one SNP call.

3. We also discard clusters whose sequences align to any of the completely masked sequences in the earlier steps. This includes reads that have local repeats as determined earlier and reads that had less than 40 unmasked bp. The same LASTZ parameters and score-sets used to calculate overlaps between the reads, are also used for aligning the cluster sequences to the repeats. This is done to ameliorate any local frequency bias generated in a particular sequencing run. An additional step for shorter illumina reads removes reads from the clusters if the cluster corresponding to the read has been termed as a candidate repeat.

4. Next we now create “micro-assemblies”, i.e., de novo assemblies for each selected cluster. For the 454 technology we extract the reads corresponding to the members of a cluster from the original SFF file to create an input SFF file for Newbler, which is part of the software package distributed by Roche/454 with their sequencing machines. We use Velvet [40] for assembling the Illumina short reads. A different assembler can be specified, as long as it works with the particular sequencing technology and generates output in the ACE file format [50].

5. We then read the ACE file output and deduce positions in the assembly where two different alleles are observed with support from at least two reads for each allele.

6. The final step filters the candidate locations to find SNPs which are suitable for genotyping. The filters involve a combination of factors including the number of variants and reads found in the assembled contig, the distance of the candidate SNP from either end of the contig and the quality score for reads. There is also a special filter to disallow SNPs in homopolymer regions. In the reported datasets (sets generated by 454 and Illumina) the following requirements were imposed:

   (a) Only one SNP is found in a micro-assembled sequence.

   (b) The SNP has at least 50 bp on both sides in the assembled sequence for the 454 datasets, and at least 40 bp on both sides for the Illumina reads.

   (c) The minimum quality score for the reads at the site is 20. Reads supporting an allele with a lower quality score are not counted.

   (d) The SNP is not part of a homopolymer of length greater than 4 bp in a window of length 13 bp centered on the SNP.

4.3 Case Studies

We are applying this strategy to the Tasmanian devil (*Sarcophilus harrisi*), the largest living carnivorous marsupial. In wild, it is only found in the state of Tasmania, Australia. The species is under threat of extinction by an aggressive transmittable cancer [56]. It has been protected by law since 1941 and efforts are underway for its conservation. One of the suggested steps is the formation of an insurance population for the species. The selection of the founding members is an important decision, with a view to maintain variability in the subsequent generations. There have been recent reports suggesting a positive relationship between heterozygosity and breeding success in endangered species, as well as heterozygosity and sperm quality in endangered mammals [67]. This necessitates the use of genomic sequence in the design of the breeding program. We do not

have a complete genomic sequence of the Devil and the closest available genome sequence is from Opossum (*Monodelphis domestica*), which diverged from the Sarcophilus about 100 million years ago.

We decided to use this pipeline on sequences from two Tasmanian devils to predict the locations of variability. The sequenced individuals were selected from the two extreme geographical locations of the island, and for immunological characteristics. Using just 0.5X data from one individual and 0.3X data from another individual, we were able to predict enough high quality SNPs for a custom genotyping array. The results from our pipeline were then verified in 168 individuals with a success rate of more than 80%. A SNP is termed a success if it can be used to identify single nucleotide variations among geographically distinct animals. Since then, we have had more sequence from both the individuals and have predicted 5015 high quality SNPs. The filtering criterion and parameters for this dataset were the same as described for the Watson genome dataset.

Chapter 5

Conclusions and Future Work

We have described some of the problems related to genome-assembly and how our tools address those issues. DIAL tries to convert the whole-genome problem into a local problem, to call SNPs suitable for further experimental analysis. We believe that further theoretical analysis would be necessary to improve its performance. A new module in DIAL reduces the disk I/O by creating an index, which contains the name and exact location of every read. This enables it to jump to the exact location when it extracts the details of the read from the SFF files, rather than having to iterate through separate files multiple times.

We have presented YASRA, a comparative assembler and demonstrated that it performs better than the existing tools under certain circumstances. In the next few sections, we discuss some of the possible improvements to it. We discuss the concept of “hybrid assemblies” and their importance in future projects. We discuss some simple suggested improvements to YASRA, especially concentrating on “assembly validation”, a field which will continue to grow as more and more genomes get sequenced.

5.1 Hybrid Assemblies

“Hybrid assembly” refers to an assembly where the reads come from different sequencing methods. A lot of assemblers such as ARACHNE support hybrid assemblies to a limited extent. Most of the assemblers prefer to use Sanger reads as the backbone of the assembly, to which other types of reads are added. In essence, the short reads are used for increasing coverage depth, to capture any local variation, whereas the Sanger reads are used to form the skeleton contigs. But there are other benefits to combining reads from sequencing technologies. For example, Illumina paired-end reads can be used to generate scaffolds for the 454 fragment generated contigs. Illumina reads can be used for correcting the homopolymer-run errors that plague 454 reads. A combination of

reads from several different sequencing technologies might also help to overcome any significant bias in reads from an single sequencing technology. Attempts at combining 454 and Solexa reads have met with limited success [68, 69]. Some of the factors make hybrid assemblies difficult are:

1. Reads from different sequencing technologies have different error profiles, and software that makes the best use of one does not necessarily do well with the other.

2. Different set of assumptions are required for different sequencing technologies. For example, vector trimming of reads makes no sense for reads that came from some pyrosequencing machine, since there is no cloning involved.

3. Most assemblers use expected coverage to disambiguate repeats, however the biases of the NGS platforms are still not very well understood. So that information cannot be used judiciously.

We have used the tools developed by the author in conjunction with some third-party tools to construct the assembly of SAR11 (*Pelagibacter ubique*), which is the numerically dominant microorganism in the ocean surface. SAR11 undergoes regular seasonal cycles but the factors that control SAR11 populations, and other bacterioplankton populations, are largely unknown. The motive of the study is to identify some of these factors, which would assist in formulation of predictive models for planktonic ecosystems. The steps taken to assemble the various reads for this dataset can be enumerated as follows:

1. We use Newbler [10] to create a de novo assembly of the reads generated using 454 technology.

2. We use Velvet [40] to create a de novo assembly of the reads generated using Solexa technology using a k-mer value of 21 and a cut-off value of 3.0.

3. We use the Newbler mapping assembler to generate contigs using the whole genome sequence of *Candidatus Pelagibacter ubique* HTCC1002 as the reference.

4. We break the generated contigs into 500 bp intervals, where adjacent intervals overlap each other by 100 bp.

5. We now use YASRA to combine the pseudo-reads generated from the three separate assemblies. The whole-genome sequence of *Candidatus Pelagibacter ubique* HTCC1002 was used as a reference.

6. We then use the YASRA contigs as the backbone to align all the 454 and Solexa reads, creating the final assembly.

This resulted in 66 contigs, spanning 1,287,422 bases, with a N50 (the length x such that 50% of the genome lies in blocks of x or longer) value of 43,317 bp. The length of the assembled genome is approximately 96.97% of the expected size of the genome. This approach uses data from 454 reads as well as Illumina reads, utilizing both comparative and de novo assembly to generate the consensus sequence. We believe that this will be the key to success in the future. Balancing quality and cost will entail combining data from several different sources and the hybrid assembly approach definitely needs to be explored further. The k-mer approach especially seems suitable for such an endeavor, as the read lengths become immaterial as soon as the de Bruijn graph is built.

5.2 YASRA

YASRA was primarily designed to work with sequences generated using Roche/454 family of instruments. It was then modified to handle shorter Illumina reads. Since then, it has proved its utility in several projects, and the author intends to maintain and develop the software further. Some of the suggested improvements for the software include:

1. A scaffolding module to order and orient the contigs is necessary, specially as the mate-pair protocol in 454 reads matures and becomes more and more common.

2. YASRA calculates the tiling path for the reads using Prim's algorithm. The current implementation uses a simple binary heap data structure and an adjacency list representation, running in time $O(E \, log \, V)$. A more sophisticated Fibonacci heap implementation can bring the time complexity down to $O(E + V \; log \; V)$. Even though the current implementation is sufficient for use with 454 reads, we need to improve performance to handle an increasing number of shorter Illumina reads.

3. YASRA contains a error-checking module, but it only checks for low coverage regions and regions where the number of rejected reads exceed the number of hits. Such regions represent an error in the assembly and the contig is split into separate fragments at that location. However many other heuristics can be used to validate the assembly and we discuss some of them in the next section.

4. Support for SOLiD reads in color-space.

5. YASRA has scripts to convert the output into ACE format which can then be viewed using Hawkeye or Consed. However, there are other viewers like EagleView, which are better suited for the large number of reads in an assembly, which need to be supported by YASRA.

6. The analysis in YASRA is maintained using Makefiles and there are no options to add more runs to an existing assembly. Some sort of project management needs to be added to facilitate use of YASRA in larger projects.

5.3 Assembly Validation

The vast amounts of data generated using NGS technologies, and the resulting assemblies can no longer be validated “by-hand”; something that was possible a few years ago. Considering that these assemblies are the backbone for all downstream analysis, we believe that more time and effort should be spent on designing algorithms and tools which can validate them.

The source of most mis-assemblies are “repeats”. Such regions can be identified using statistical approaches: either an ad hoc probability distribution based on *k*-mer frequencies [70] or poisson statistics on the read depth [40, 20]. This strategy is sufficient for reads generated with 454 technology, and it can be applied to Illumina reads with a small correction to account for the GC content influence as reported in Bentley et al. [71]. When identified, mate-pairs should be used to disambiguate these repeats into multiple repeat copies. Singleton reads (reads that are not part of the assembly) should also be realigned to the assembled sequence to check if portions of the read align well with different regions in the assembly. If this pattern is observed for a number of reads at the same location, that should be a cause for concern. It could signal a collapse of some repeat region or some other mis-assembly.

Another source of mis-assemblies is the diploid nature of the large mammalian genomes. The two copies of a chromosome are always slightly divergent, and that can lead to creation of two separate contigs for the same region of the genome. There are several popular tools [72, 73] that can help in reconstruction of the two haplotypes from an assembled sequence. We believe that these tools should be used in conjunction with a post-processor which uses the coverage information, the mate-pair information and the overlap information for the contigs, to make the haplotype assembly of the individual.

SOLiD data uses “2-base encoding” to reduce their error rate. Even though the paradigm leads to lower coverage in regions of high SNP density, we believe that these reads can be used to validate assemblies, specially the ones generated using 454 reads. In the default SOLiD pipeline, the reads are mapped to an existing reference and the generated nucleotide consensus remains dependent on the reference sequence till the end. An alternate approach can be enumerated as follows:

1. Map the SOLiD reads using the assembled contigs as the reference.

2. Remove the reference from consideration and assemble the reads in color-space.

3. Map the generated contigs to the assembled contigs again in color-space.

4. Dinucleotides AA,CC,GG,TT are all represented as “0” in color-space. The number of consecutive “0”’s signify the length of a homopolymer and this information can now be used to correct homopolymer-run errors in the assembled contigs.

Bibliography

[1] **Genome 10K sample collection project.** [http://genome10k.soe.ucsc.edu/].

[2] **1000 genomes: a deep catalog of human genetic variation.** [http://www.1000genomes.org/page.php].

[3] **Time to sequence the 'red and the dead'.** [http://dx.doi.org/10.1038/458812a].

[4] Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ: **Basic local alignment search tool**. *Journal of molecular biology* 1990, **215**:403–410.

[5] Kent WJ: **BLAT-the BLAST-like alignment tool**. *Genome Res.* 2002, **12**:656–664.

[6] Sanger F, Nicklen S, Coulson AR: **DNA sequencing with chain terminating inhibitors**. *Proceedings of the National Academy of Sciences of the United States of America* 1997, **74**:5463–5467.

[7] **Advanced sequencing technology awards 2004.** [http://www.genome.gov/12513162].

[8] **Illumina personal genome service.** [http://www.everygenome.com/].

[9] Garber M, Zody MC, Arachchi HM, Berlin A, Gnerre S, Green LM, Lennon N, Nusbaum C: **Closing gaps in the human genome using sequencing by synthesis**. *Genome Biology* 2009, **10**:R60.

[10] Margulies M, Egholm M, Altman WE, Attiya S, Bader JS, Bemben LA, Berka J, Braverman MS, Chen Y, Chen Z, Dewell SB, Du L, Fierro JM, Gomes XV, Godwin BC, He W, Helgesen S, Ho CH, Irzyk GP, Jando SC, Alenquer MLI, Jarvie TP, Jirage KB, Kim J, Knight JR, Lanza JR, Leamon JH, Lefkowitz SM, Lei M, Li J, Lohman KL, Lu H, Makhijani VB, McDade KE, McKenna MP, Myers EW, Nickerson E, Nobile JR, Plant R, Puc BP, Ronan MT, Roth GT, Sarkis GJ, Simons JF, Simpson JW, Srinivasan M, Tartaro KR, Tomasz A, Vogt KA, Volkmer

GA, Wang SH, Wang Y, Weiner MP, Yu P, Begley RF, Rothberg JM: **Genome sequencing in microfabricated high-density picolitre reactors**. *Nature* 2005, **437**:376–380.

[11] Bennett S: **Solexa ltd**. *Pharmacogenomics* 2004, **5**:433–438.

[12] Pushkarev D, Neff NF, Quake SR: **Single-molecule sequencing of an individual human genome**. *Nat Biotech* 2009, [http://dx.doi.org/10.1038/nbt.1561].

[13] Clarke J, Wu HC, Jayasinghe L, Patel A, Reid S, Bayley H: **Continuous base identification for single-molecule nanopore DNA sequencing**. *Nat Nano* 2009, **4**:265–270.

[14] Pop M: **Shotgun sequence assembly**. *Advances in Computers* 2004, **60**:193–248.

[15] **PHRAP.**[http://phrap.org].

[16] Sutton GG, White O, Adams MD, Kerlavage AR: **TIGR assembler: a new tool for assembling large shotgun sequencing projects**. *Genome Science and Technology* 1995, **1**:9–19.

[17] Huang X: **A contig assembly program based on sensitive detection of fragment overlaps**. *Genomics* 1992, **12**:18–25.

[18] Huang X: **An improved sequence assembly program**. *Genomics* 1996, **33**:21–31.

[19] Teng S, Yao F: **Approximating shortest superstrings**. In *IEEE Symposium on Foundations of Computer Science* 1993:158–165.

[20] Myers EW, Sutton GG, Delcher AL, Dew IM, Fasulo DP, Flanigan MJ, Kravitz SA, Mobarry CM, Reinert K, Remington KA, Anson EL, Bolanos RA, Chou HH, Jordan CM, Halpern AL, Lonardi S, Beasley EM, Brandon RC, Chen L, Dunn PJ, Lai Z, Liang Y, Nusskern DR, Zhan M, Zhang Q, Zheng X, Rubin GM, Adams MD, Venter JC: **A whole-genome assembly of Drosophila**. *Science* 2000, **287**:2196–2204.

[21] Batzoglou S, Jaffe DB, Stanley K, Butler J, Gnerre S, Mauceli E, Berger B, Mesirov JP, Lander ES: **ARACHNE: a whole-genome shotgun assembler**. *Genome Research* 2002, **12**:177–189.

[22] Drmanac R, Drmanac S, Chui G, Diaz R, Hou A, Jin H, Jin P, Kwon S, Lacy S, Moeur B, Shafto J, Swanson D, Ukrainczyk T, Xu C, Little D: **Sequencing by hybridization (SBH): advantages, achievements, and opportunities.** *Adv Biochem Eng Biotechnol* 2002, **77**:75–101.

[23] Idury RM, Waterman MS: **A new algorithm for DNA sequence assembly**. *Journal of Computational Biology* 1995, **2**:291–306.

[24] Kent WJ, Haussler D: **Assembly of the working draft of the human genome with GigAssembler**. *Genome Research* 2001, **11**:1541–1548.

[25] Li H, Ruan J, Durbin R: **Mapping short DNA sequencing reads and calling variants using mapping quality scores**. *Genome Research* 2008, **18**:1851–1858.

[26] Buhler J: **Efficient large-scale sequence comparison by locality-sensitive hashing**. *Bioinformatics* 2001, **17**:419–428.

[27] Ferragina P, Manzini G: **Opportunistic data structures with applications**. In *Proceedings of the 41st Symposium on Foundations of Computer Science* 2000:390–398.

[28] Burrows M, Wheeler DJ: **A block-sorting lossless data compression algorithm**. Tech. rep., Digital SRC Research Report 1994.

[29] Read TD, Salzberg SL, Pop M: **Comparative genome sequencing for discovery of novel polymorphisms in *Bacillus antracis***. *Science* 2002, **296**:2028–2033.

[30] Gilbert MTP, Tomsho LP, Rendulic S, Packard M, Drautz DI, Sher A, Tikhonov A, Dalen L, Kuznetsova T, Kosintsev P, Campos PF, Higham T, Collins MJ, Wilson AS, Shidlovskiy F, Buigues B, Ericson PGP, Germonpre M, Gotherstrom A, Iacumin P, Nikolaev V, Nowak-Kemp M, Willerslev E, Knight JR, Irzyk GP, Perbost CS, Fredrikson KM, Harkins TT, Sheridan S, Miller W, Schuster SC: **Whole-genome shotgun sequencing of mitochondria from ancient hair shafts**. *Science* 2007, **317**:1927–1930.

[31] Herring CD, Raghunathan A, Honisch C, Patel T, Applebee KM, Joyce AR, Albert TJ, Blattner FR, van den Boom D, Cantor CR, Bernhard: **Comparative genome sequencing of *Escherichia coli* allows observation of bacterial evolution on a laboratory timescale**. *Nature Genetics* 2006, **38**:1406–1412.

[32] Pop M, Phillippy A, Delcher AL, Salzberg SL: **Comparative genome assembly**. *Briefings in Bioinformatics* 2004, **5**:237–248.

[33] Kurtz S, Phillippy A, Delcher AL: **Versatile and open software for comparing large genomes**. *Genome Biology* 2004, **5**:R12.

[34] Pop M, Kosack SD, Salzberg SL: **Hierarchical scaffolding with Bambus**. *Genome Research* 2004, **14**:149–159.

[35] Dohm JC, Lottaz C, Borodina T, Himmelbauer H: **SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing**. *Genome Research* 2007, **17**:1697–1706.

[36] Warren RL, Sutton GG, Jones SJM, Holt RA: **Assembling millions of short DNA sequences using SSAKE**. *Bioinformatics* 2007, **23**:500–501.

[37] Jeck WR, Reinhardt JA, Baltrus DA, Hickenbotham MT, Magrini V, Mardis ER, Dangl JL, Jones CD: **Extending assembly of short DNA sequences to handle error**. *Bioinformatics* 2007, **23**:2942–2944.

[38] Bryant D, Wong WK, Mockler T: **QSRA - a quality-value guided de novo short read assembler**. *BMC Bioinformatics* 2009, **10**:69.

[39] Hernandez D, Franois P, Farinelli L, ⊘ster$\mathring{a}$s M, Schrenzel J: **De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer**. *Genome Research* 2008, **18**:802–809.

[40] Zerbino DR, Birney E: **Velvet: algorithms for de novo short read assembly using de Bruijn graphs**. *Genome Research* 2008, **18**:821–829.

[41] Butler J, MacCallum I, Kleber M, Shlyakhter IA, Belmonte MK, Lander ES, Nusbaum C, Jaffe DB: **ALLPATHS: de novo assembly of whole-genome shotgun microreads**. *Genome Research* 2008, **18**:810–820.

[42] Chaisson MJ, Pevzner PA: **Short read fragment assembly of bacterial genomes**. *Genome Research* 2008, **18**:324–330.

[43] Pevzner PA, Tang H, Waterman MS: **An eulerian path approach to DNA fragment assembly**. *Proceedings of the National Academy of Sciences of the United States of America* 2001, **98**:9748–9753.

[44] Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJ, Birol I: **ABySS: a parallel assembler for short read sequence data**. *Genome Research* 2009, **19**:1117–1123.

[45] Schmidt B, Sinha R, Beresford-Smith B, Puglisi SJ: **A fast hybrid short read fragment assembly algorithm**. *Bioinformatics* 2009, **25**:2279–2280.

[46] Krause J, Dear PH, Pollack JL, Slatkin M, Spriggs H, Barnes I, Lister AM, Ebersberger I, Paabo S, Hofreiter M: **Multiplex amplification of the mammoth mitochondrial genome and the evolution of Elephantidae**. *Nature* 2006, **439**:724–727.

[47] Harris RS: **Improved pairwise alignment of genomic DNA**. *PhD thesis*, Penn State University, Computer Science and Engineering 2007.

[48] Cormen TH, Leiserson CE, Rivest C, Stein: *Introduction to algorithms*. MIT Press 2001.

[49] Anson EL, Myers EW: **ReAligner: a program for refining DNA sequence multi-alignments**. *Journal of Computational Biology* 1997, **4**:369–383.

[50] Gordon D, Abajian C, Green P: **Consed: a graphical tool for sequence finishing**. *Genome Research* 1998, **8**:195–202.

[51] Schatz MC, Phillippy AM, Shneiderman B, Salzberg SL: **Hawkeye: a visual analytics tool for genome assemblies**. *Genome Biology* 2007, **8**:R34.

[52] Gilbert MTP, Drautz DI, Lesk AM, Ho SYW, Qi J, Ratan A, Hsu CH, Sher A, Daln L, Gtherstrm A, Tomsho LP, Rendulic S, Packard M, Campos PF, Kuznetsova TV, Shidlovskiy F, Tikhonov A, Willerslev E, Iacumin P, Buigues B, Ericson PGP, Germonpr M, Kosintsev P, Nikolaev V, Nowak-Kemp M, Knight JR, Irzyk GP, Perbost CS, Fredrikson KM, Harkins TT, Sheridan S, Miller W, Schuster SC: **Intraspecific phylogenetic analysis of Siberian woolly mammoths using complete mitochondrial genomes**. *Proceedings of the National Academy of Sciences* 2008, **105**:8327–8332.

[53] Miller W, Drautz DI, Janecka JE, Lesk AM, Ratan A, Tomsho LP, Packard M, Zhang Y, McClellan LR, Qi J, Zhao F, Gilbert MTP, Daln L, Arsuaga JL, Ericson PG, Huson DH, Helgen KM, Murphy WJ, Gtherstrm A, Schuster SC: **The mitochondrial genome sequence of the Tasmanian tiger (*Thylacinus cynocephalus*)**. *Genome Research* 2009, **19**:213–220.

[54] Willerslev E, Gilbert MT, Binladen J, Ho S, Campos P, Ratan A, Tomsho L, da Fonseca R, Sher A, Kuznetsova T, Nowak-Kemp M, Roth T, Miller W, Schuster S: **Analysis of complete mitochondrial genomes from extinct and extant rhinoceroses reveals lack of phylogenetic resolution**. *BMC Evolutionary Biology* 2009, **9**:95.

[55] Lindqvist C, Schuster S, Sun Y, Talbot S, Qi J, Ratan A, Tomsho L, McClellan L, Zhang Y, Zeyl E, Ehrich D, Aars J, Miller W, Ingolfsson O, Bachmann L, Wiig O: **Complete mitochondrial genome of a Pleistocene jawbone unveils the origin of polar bear**. [To appear].

[56] Miller W, Wittekindt N, Petersen DC, Qi J, Ratan A, Woodbridge P, Tomsho LP, McClellan LR, Tindall EA, Kreiss A, Hardie R, Pycroft S, Bertelsen MF, Dixon D, Murphy WJ, Helgen KM, Fleischer R, Lesk A, Pringle1 T, Woods G, Jones M, Zhang Y, Harkins TT, Schuster SC, Hayes VM: **DeversiTyping the Tasmanian devil: a strategy for insurance populations for endangered species**. [To appear].

[57] **Gymnosperms on the tree of life** [http://www.huh.harvard.edu/research/mathews-lab/tolHtmlSite/index.htm].

[58] **Devin Locke**. *Personal Communication*.

[59] Schuster SC, Miller W, Ratan A, Tomsho LP, Giardine B, Lindsay RK, Harris RS, Petersen DC, Zhao F, Qi J, Alkan C, Sun Y, Drautz D, Bouffard P, Burhans R, Riemer C, Wittekindt NE, Moorjani P, Tindall EA, Taliwal V, Danko C, Teo WS, Buboltz A, Oosthiusen A, Steenkamp A, Oosthuizen H, Venter P, Gajewski J, Makova K, Nekrutenko A, Mardis E, Patterson N, Pringle T, Chiaromonte F, Mulliken J, Eichler EE, Hardison RC, Gibbs R, Siepel A, Harkins TT, Hayes VM: **Complete genomes of southern African hunterGatherers and Archbishop Desmond Tutu**. [To appear].

[60] **Human1M-duo beadchip** [http://www.illumina.com/pages.ilmn?ID=261].

[61] Wheeler DA, Srinivasan M, Egholm M, Shen Y, Chen L, McGuire A, He W, Chen Y, Makhijani V, Roth GT, Gomes X, Tartaro K, Niazi F, Turcotte CL, Irzyk GP, Lupski JR, Chinault C, Song X, Liu Y, Yuan Y, Nazareth L, Qin X, Muzny DM, Margulies M: **The complete genome of an individual by massively parallel DNA sequencing**. *Nature* 2008, **452**:872–876.

[62] Tishkoff SA, Reed FA, Friedlaender FR, Ehret C, Ranciaro A, Froment A, Hirbo JB, Awomoyi AA, Bodo JM, Doumbo O, Ibrahim M, Juma AT, Kotze MJ, Lema G, Moore JH, Mortensen H, Nyambo TB, Omar SA, Powell K, Pretorius GS, Smith MW, Thera MA, Wambebe C, Weber JL, Williams SM: **The genetic structure and history of Africans and African Americans**. *Science* 2009, **324**:1035–1044.

[63] Ratan A, Zhang Y, Hayes VM, Schuster SC, Miller W: **Calling SNPs without a reference sequence**. [To appear].

[64] Ahn SM, Kim TH, Lee S, Kim D, Ghang H, Kim DS, Kim BC, Kim SY, Kim WY, Kim C, Park D, Lee YS, Kim S, Reja R, Jho S, Kim CG, Cha JY, Kim KH, Lee B, Bhak J, Kim SJ: **The first Korean genome sequence and analysis: full genome sequencing for a socio-ethnic group**. *Genome Research* 2009, [http://genome.cshlp.org/content/early/2009/06/24/gr.092197.109.abstract].

[65] Rivest RL: **The MD5 message-digest algorithm**. *Internet Request for Comments* 1992. [RFC 1321].

[66] Morgulis A, Gertz EM, Schffer AA, Agarwala R: **WindowMasker: window-based masker for sequenced genomes**. *Bioinformatics* 2006, **22**:134–141.

[67] Fitzpatrick JL, Evans JP: **Reduced heterozygosity impairs sperm quality in endangered mammals**. *Biology Letters* 2009, **5**:320–323.

[68] Reinhardt JA, Baltrus DA, Nishimura MT, Jeck WR, Jones CD, Dangl JL: **De novo assembly using low-coverage short read sequence data from the rice pathogen *Pseudomonas syringae* pv. *oryzae***. *Genome Research* 2009, **19**:294–305.

[69] Rodrigue S, Malmstrom RR, Berlin AM, Birren BW, Henn MR, Chisholm SW: **Whole genome amplification and de novo assembly of single bacterial cells**. *PLoS ONE* 2009, **4**:e6864.

[70] Kim S, Liao L, Tomb JF: **A probabilistic approach to sequence assembly validation**. In *BIOKDD* 2001:38–43.

[71] Bentley DR, Balasubramanian S, Swerdlow HP, Smith GP, Milton J, Brown CG, Hall KP, Evers DJ, Barnes CL, Bignell HR, Boutell JM, Bryant J, Carter RJ, Keira Cheetham R, Cox AJ, Ellis DJ, Flatbush MR, Gormley NA, Humphray SJ, Irving LJ, Karbelashvili MS, Kirk SM, Li H, Liu X, Maisinger KS, Murray LJ, Obradovic B, Ost T, Parkinson ML, Pratt MR, Rasolonjatovo IM, Reed MT, Rigatti R, Rodighiero C, Ross MT, Sabot A, Sankar SV, Scally A, Schroth GP, Smith ME, Smith VP, Spiridou A, Torrance PE, Tzonev SS, Vermaas EH, Walter K, Wu X, Zhang L, Alam MD, Anastasi C, Aniebo IC, Bailey DM, Bancarz IR, Banerjee S, Barbour SG, Baybayan PA, Benoit VA, Benson KF, Bevis C, Black PJ, Boodhun A, Brennan JS, Bridgham JA, Brown RC, Brown AA, Buermann DH, Bundu AA, Burrows JC, Carter NP, Castillo N, Chiara E Catenazzi M, Chang S, Neil Cooley R, Crake NR, Dada OO, Diakoumakos KD, Dominguez-Fernandez B, Earnshaw DJ, Egbujor UC, Elmore DW, Etchin SS, Ewan MR, Fedurco M, Fraser LJ, Fuentes Fajardo KV, Scott Furey W, George D, Gietzen KJ, Goddard CP, Golda GS, Granieri PA, Green DE, Gustafson DL, Hansen NF, Harnish K, Haudenschild CD, Heyer NI, Hims MM, Ho JT, Horgan AM, Hoschler K, Hurwitz S, Ivanov DV, Johnson MQ, James T, Huw Jones TA, Kang GDD, Kerelska TH, Kersey AD, Khrebtukova I, Kindwall AP, Kingsbury Z, Kokko-Gonzales PI, Kumar A, Laurent MA, Lawley CT, Lee SE, Lee X, Liao AK, Loch JA, Lok M, Luo S, Mammen RM, Martin JW, McCauley PG, McNitt P, Mehta P, Moon KW, Mullens JW, Newington T, Ning Z, Ling Ng B, Novo SM, O'Neill MJ, Osborne MA, Osnowski A, Ostadan O, Paraschos LL, Pickering L, Pike AC, Pike AC, Chris Pinkard D, Pliskin DP, Podhasky J, Quijano VJ, Raczy C, Rae VH, Rawlings SR, Chiva Rodriguez A, Roe PM, Rogers J, Rogert Bacigalupo MC, Romanov N, Romieu A, Roth RK, Rourke NJ, Ruediger ST, Rusman E, Sanches-Kuiper RM, Schenker MR, Seoane JM, Shaw RJ, Shiver MK, Short SW, Sizto NL, Sluis JP, Smith MA, Ernest Sohna Sohna J, Spence EJ, Stevens K, Sutton N, Szajkowski L, Tregidgo CL, Turcatti G, Vandevondele S, Verhovsky Y,

Virk SM, Wakelin S, Walcott GC, Wang J, Worsley GJ, Yan J, Yau L, Zuerlein M, Rogers J, Mullikin JC, Hurles ME, McCooke NJ, West JS, Oaks FL, Lundberg PL, Klenerman D, Durbin R, Smith AJ: **Accurate whole human genome sequencing using reversible terminator chemistry.** *Nature* 2008, **456**:53–59.

[72] Bansal V, Bafna V: **HapCUT: an efficient and accurate algorithm for the haplotype assembly problem**. *Bioinformatics* 2008, **24**:i153–159.

[73] Bansal V, Halpern AL, Axelrod N, Bafna V: **An MCMC algorithm for haplotype assembly from whole-genome sequence data**. *Genome Research* 2008, **18**:1336–1346.

[74] Garey MR, Johnson DS: *Computers and Intractability*. W.H. Freeman, New York 1979.

[75] Peltola H, Soderlund H, Tarhio J, Ukkonen E: **Algorithms for some string matching problems arising in molecular genetics**. In *IFIP Congress* 1983:59–64.

[76] Teng S, Yao FF: **Approximating shortest superstrings**. *SIAM Journal on Computing* 1997, **26**:410–417.

[77] Armen C, Stein C: **A 2 2/3-approximation algorithm for the shortest superstring problem**. In *Proc. Combinatorial Pattern Matching* 1996.

[78] Kececioglu JD: **Exact and approximation algorithms for DNA sequence reconstruction**. Tech. rep., University of Arizona 1991.

[79] Huang X, Madan A: **CAP3: a DNA sequence assembly program**. *Genome Research* 1999, **9**:868–877.

[80] Huang X, Wang J, Aluru S, Yang S, Hillier L: **PCAP: a whole-genome assembly program**. *Genome Research* 2003, **13**:2164–2170.

[81] Smith TF, Waterman MS: **Identification of common molecular subsequences**. *Journal of Molecular Biology* 1981, **147**:195–197.

[82] Churchill GA, Waterman MS: **The accuracy of DNA sequences: estimating sequence quality**. *Genomics* 1992, **14**:89–98.

[83] Myers EW: **Towards simplifying and accurately formulating fragment assembly**. *Journal of Computational Biology* 1995, **2**:275–290.

[84] Jaffe DB, Butler J, Gnerre S, Mauceli E, Lindblad-Toh K, P MJ, Zody MC, Lander ES: **Whole-genome sequence assembly for mammalian genomes: Arachne 2**. *Genome Research* 2003, **13**:91–96.

[85] Pe'er I, Shamir R: **Spectrum alignment: efficient resequencing by hybridization**. In *Proceedings of the Eight International Conference on Intelligent Systems for Molecular Biology (ISMB)* 2000.

[86] Parsons RJ, Forrest S, Burks C: **Genetic algorithms, operators and DNA fragment assembly**. *Machine Learning* 1995, **21**:11–33.

[87] Goldberg MK, Lim DT: **A learning algorithm for the shortest superstring problem**. In *Proceedings of the Atlantic Symposium on Computational Biology and Genome Information Systems and Technology, Durham, NC* 2001.

[88] Ewing B, Hillier L, Wendl MC, Green P: **Base-calling of automated sequencer traces using Phred I accuracy assessment**. *Genome Research* 1998, **8**:175–185.

[89] Ewing B, Green P: **Base-calling of automated sequencer traces using Phred II error probabilities**. *Genome Research* 1998, **8**:186–194.

[90] Rumble SM, Lacroute P, Dalca AV, Fiume M, Sidow A, Brudno M: **SHRiMP: accurate mapping of short color-space reads**. *PLoS Comput Biol* 2009, **5**:e1000386.

[91] Rasmussen KR, Stoye J, Myers EW: **Efficient q-gram filters for finding all ϵ-matches over a given length**. *Journal of Computational Biology* 2006, **13**:296–308.

[92] Smith A, Xuan Z, Zhang M: **Using quality scores and longer reads improves accuracy of Solexa read mapping**. *BMC Bioinformatics* 2008, **9**:128.

[93] Schatz MC: **CloudBurst: highly sensitive read mapping with MapReduce**. *Bioinformatics* 2009, **25**:1363–1369.

[94] Lin H, Zhang Z, Zhang MQ, Ma B, Li M: **ZOOM! zillions of oligos mapped**. *Bioinformatics* 2008, **24**:2431–2437.

[95] Choi KP, Zeng F, Zhang L: **Good spaced seeds for homology search**. *Bioinformatics* 2004, **20**:1053–1059.

[96] Jiang H, Wong WH: **SeqMap: mapping massive amount of oligonucleotides to the genome**. *Bioinformatics* 2008, **24**:2395–2396.

[97] Weese D, Emde AK, Rausch T, Dring A, Reinert K: **RazerS-fast read mapping with sensitivity control**. *Genome Research* 2009, **19**:1646–1654.

[98] Doring A, Weese D, Rausch T, Reinert K: **SeqAn: an efficient, generic C++ library for sequence analysis**. *BMC Bioinformatics* 2008, **9**:11.

[99] Li R, Li Y, Kristiansen K, Wang J: **SOAP: short oligonucleotide alignment program**. *Bioinformatics* 2008, **24**:713–714.

[100] **Novocraft aligner** [http://www.novocraft.com].

[101] **MOSAIK genome aligner** [http://bioinformatics.bc.edu/marthlab/Mosaik].

[102] Homer N, Nelson SF, Merriman B: **BFAST**[http://genome.ucla.edu/bfast]. [Unpublished].

[103] Eaves HL, Gao Y: **MOM: maximum oligonucleotide mapping**. *Bioinformatics* 2009, **25**:969–970.

[104] Campagna D, Albiero A, Bilardi A, Caniato E, Forcato C, Manavski S, Vitulo N, Valle G: **PASS: a program to align short sequences**. *Bioinformatics* 2009, **25**:967–968.

[105] Li R, Yu C, Li Y, Lam TW, Yiu SM, Kristiansen K, Wang J: **SOAP2: an improved ultrafast tool for short read alignment**. *Bioinformatics* 2009, **25**:1966–1967.

[106] Li H, Durbin R: **Fast and accurate short read alignment with Burrows-Wheeler transform**. *Bioinformatics* 2009, **25**:1754–1760.

[107] Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, Subgroup GPDP: **The sequence alignment/map format and SAMtools**. *Bioinformatics* 2009, **25**:2078–2079.

[108] Langmead B, Trapnell C, Pop M, Salzberg SL: **Ultrafast and memory-efficient alignment of short DNA sequences to the human genome**. *Genome Biology* 2009, **10**.

Appendix A

Assembly Paradigms

We elaborate on the various approaches to the assembly problem in this section.

A.1 Shortest Superstring

Initial attempts at solving the assembly problem were based on a simple theoretical model: the superstring problem. Within the realm of this model, the assembly problem can be posed as follows:

> Given the set of reads $f_1, f_2, \ldots, f_n$, find the shortest string that contains the original reads as substrings.

In its general form this problem has been proven to be NP-hard [74]. A greedy algorithm [75] was proven to yield a solution that is at most 4 times longer than the shortest superstring. A simple improvement to this could yield results which are 3 times longer than the optimal string. Other attempts [76, 77] have proposed closer approximation schemes, but there are certain flaws with working within the framework of this model itself. This model conveniently ignores the problems of assembly with repeats, which collapse if the minimal superstring is found. Another problem pertains to the assumption that the reads themselves are error-free. This assumption is certainly flawed with the sequencing technologies that exist today. So, it is actually better to restate the assembly problem as follows:

> Given the set of reads $f_1, f_2, \ldots, f_n$, find the shortest string S, such that each read matches a substring of S with less than ϵ errors.

This problem has also been proven to be NP-complete [78]. Despite the obvious shortcomings, this framework has been used successfully in a number of greedy assembly algorithms [17, 18, 79, 80, 15, 16]. The major steps involved can be enumerated as follows:

1. Add all the reads to a common store as singleton contigs.

2. Find the two contigs with the best “overlap”. Two reads are said to overlap if they align, such that the only differences between them can be explained by sequencing or cloning errors. The differences, or the edit-distance between the reads, are identified by standard alignment algorithms [81] and only alignments with low error rates are used. Furthermore, alignments must be “proper”. This implies that alignment must start and end at sequence ends. This requirement addresses spurious overlaps caused by repeats, since repeats align with “overhangs”.

3. Repeat the above steps till no more contigs can be formed.

In its general form, the quality of the overlap is equivalent to the length of the overlap. When we consider the errors in the reads, the quality of the overlap is judged by one or more of the following:

- Length of the overlap region.
- Percentage of the bases shared by the two sequences as a fraction of the length of the overlap.
- Score of the alignment comprising of: reward for a good match, substitution penalty, gap opening penalty and gap extension penalty.
- Quality adjusted score of the alignment–the score is readjusted to take into consideration the quality scores [82] assigned by the base caller.
- Mate pairs confirming or conflicting with the overlap.

A.2 Overlap-Layout-Consensus

The original greedy approach outlined in the preceding section is local in nature, which means only reads or contigs that are to be merged are examined by the algorithm. In order to consider longer range interactions and still avoid the combinatorial explosion involved in examining all pairs of reads, Peltola et al. [75] and Kececioglu and Myers [78] introduced the overlap-layout-consensus paradigm. This model broke the assembly problem into three explicit problems, which were:

1. **Overlap**–Find all the overlaps between the reads that satisfy certain quality criterion.

2. **Layout**–Given the overlaps between the reads, find the consistent tiling of the reads that preserve most of the constraints.

3. **Consensus**–Given the tiling path of the reads determined in the prior step calculate the most probable DNA sequence, possibly assigning quality scores for each of the bases.

The overlap stage of this algorithm generates a graph whose nodes represent the reads. Two nodes are connected by an edge if they are involved in a "proper" overlap. The layout problem can now be defined as that of finding an interval in the sub-graph that maximizes a particular target function. Peltola et al. look for a layout that "uses the best possible overlaps" and Kececioglu and Myers attempt to maximize the weight of the resulting sub-graph given the weights corresponding to the quality of the overlaps. They also showed that the layout problem is NP-complete and proposed a greedy approximation algorithm for solving it. Myers [83] also introduced a variant of this problem which generates the layout that best matches the statistical characteristics of the fragment shearing process under the Kolmogorov-Smirnov statistic.

Most of the solutions proposed above start out with regions which are relatively easier to assemble and then gradually work with the areas which are harder, generally relaxing certain criterion as they go along. Myers introduced the concept of "chunk graph", where a chuck is the maximal interval sub-graph. In other words a chuck represents areas of the genome between repeats, and hence can be unambiguously resolved. Repeats cause branches in the overlap graph, signaling the end of the chunk. Chunk graph can then be augmented with additional information such as mate pair data, to reduce the complexity of the graph. This idea of chunks matured into the "unitig" concept in the Celera assembler [20]. Both the Celera assembler and Arachne [21, 84] start by identifying unambiguous areas of the genome and then use a scaffolding step to generate the final layout. Arachne, uses the mate pair information to identify "paired-pairs", that is, pairs of shotgun fragments of similar lengths whose end sequences overlap. In contrast Celera assembler uses the mate pair information in the latter stages.

A.3 Sequencing by Hybridization

A "sequencing by hybridization" experiment involves building an array of all possible k-tuple probes. The DNA sequence being sequenced is hybridized to the chip in order to find all the k-tuples present in the DNA sample. The problem is now reduced to finding the DNA strand whose k-tuple spectrum is the observed set of k-tuples. The information provided to this problem is the presence or absence of a tuple and its multiplicity in the genome. Idury and Waterman [23] proposed this framework as an alternative to the OLC paradigm. The basic idea was to convert the reads into k-tuples and then treat it as an sequencing by hybridization experiment.

In terms of graph theory, the problem can be framed as follows: Given a set of k-tuples, construct a graph G whose nodes represent all the $(k-1)$-tuples in the set. Two nodes are connected by an edge if the corresponding $(k-1)$-tuples represent the prefix and the suffix of

the original k-tuples. Thus the k-tuples are implicitly represented by the edges in the graph G. A solution to the sequencing by hybridization problem is represented by a path through the graph that visits every single edge. This is related to the Eulerian path problem of graph theory, which requires finding a path when each edge is traversed "exactly" once. However, in the SBH problem a k-mer may be used multiple times if it is involved in a repeat.

The Eulerian problem can be solved in linear time to the number of edges in the graph. However it seems obvious that sequencing errors would complicate the graph by forming spurious k-mers. And hence, when this work was extended into a practical implementation by Pevzner et al. [43], their algorithm depended heavily on an error-correction module. For a set of reads, they identify a set of "solid" k-mers, specifically those that belonged to more than a pre-determined number of reads. The error-correction can be performed by solving the spectral alignment problem [85]:

> Given a string S and a spectrum T(in this case, the set of solid k-mers), find the minimum number of mutations in S such that the spectrum of the resulting string is contained in T.

Pevzner et al. also gave an alternative heuristic approach which does not require the knowledge of the set of solid k-mers and takes advantage of the fact that errors in each reads are relatively rare. Hence they formulate the error correction problem:

> Given a set of strings S and a threshold δ, find a set of fewer than δ corrections in each read such that the number of k-mers in the spectrum of S is minimized.

The rationale behind this approach is that each sequencing error leads to k erroneous k-mers therefore increasing the size of the spectrum of S. Removing a sequencing error would thus result in a reduction of the size of the spectrum, as multiple reads contain the same base. However as the authors note, they also introduce some errors by incorrectly changing bases. They use the initial reads as a guide in generating the Eulerian path, leading to a new problem–the Eulerian Superpath problem:

> Find an Eulerian path that confirms most reads in the input.

This is extremely helpful in resolving short repeats. The authors also use the mate information as a guide to resolve the longer repeats. The remaining repeats cannot be resolved and are reported. This paradigm also forms the basis of most of the short-read de novo assemblers today.

A.4 Hierarchical Assembly

Before the successful sequencing of the Drosophila genome [20], whole genome shotgun assembly had only been done on bacterial genomes and was thought impossible for mammalian

genomes. In order to sequence large genomes, scientists followed a hierarchical approach. They broke the DNA into a collection of large fragments, with sizes between 50 and 200 kbp. These fragments were then cloned into BAC's (Bacterial Artificial Chromosome) or fosmids or cosmids. The advantages of this approach include the following:

- Small size of these fragments means that existing programs for assembly can be used.
- Small fragments ensure that there are few internal repeats.
- Fragments are long enough to provide sufficient physical markers that allow their unambiguous placement in the genome.

The steps in hierarchical assembly can be outlined as follows:

1. Scientists map the fragments to a unique location on the genome. This allows them to identify a "minimal tiling path", that is a collection of reads or fragments that covers the whole genome. Minimal tiling path is necessary in order to achieve a balance between the costs of the sequencing and its quality.
2. Each of the individual fragments in the minimum tiling path are sequenced through the shotgun method.
3. Finished fragments are assembled together using overlapping regions as guide.

As is evident from the process, we need two specialized programs for a hierarchical assembly. One of them is required to assemble within each fragment and the other one to assemble the finished fragments into a genome using a lot of the additional constraints and ancillary information. The first task can be performed by using any of the assemblers we have discussed earlier. However special programs and human intervention is required to complete the second task. The initial assembly of the human genome by the Human Genome sequencing Consortium was done using this method and required a special program GigAssembler [24].

It is important to note that hierarchical approaches are very important in the context of whole-genome shotgun sequencing. They especially becomes important in the finishing stages of a sequencing projects. Libraries with different sized reads are helpful in resolving different classes of repeats. It is important for assembly programs to allow such a hierarchical approach in order to "jump start" the assembly with the already generated contigs.

A.5 Machine Learning

One of the earliest approaches to the assembly problem was based on machine learning techniques [86, 87] which phrased the assembly problem as that of learning a string from a collection

of random substrings. Attempts at solving this paradigm have included use of genetic algorithms [86] and treatment of the algorithm as a collection of modules with a lot of parameters which can be learnt through small examples [87]. But all these machine learning based methods have never been really been tested for scalability, since the largest genome assembled with these techniques contained 177 fragments that covered 34 kbp.

Appendix B

Mapping Assemblers

The last year has seen the emergence of a lot of mapping assemblers. Here we list and briefly describe some of them. They are categorized on the basis of their approach to find the correct placement of a read.

B.1 Indexing of Reads with Hash Tables

Some of the most popular programs in this category include the following:

1. **Eland**: Eland is the first and one of the fastest aligners in this category. Even though it only works for single reads up to 32 bp length, several downstream scripts extends its functionality to include paired-end mapping and SNP calling for longer reads.

2. **MAQ**: MAQ [25] is probably the most popular of these aligners and has been the aligner of choice for a many high-profile projects like the 1000 Genomes project [2]. It indexes the first 28 bp of the reads and aligns reads with up to two mismatches in the first 28 bp, to the reference. MAQ assigns each individual alignment a phred-scaled quality score [88, 89], which estimates the probability that the true alignment is not the one found by MAQ. One of the limitations of MAQ is that it always reports a single alignment, and if a read can be aligned equally well to multiple positions, it randomly picks one position and gives it a mapping quality score of zero. The reads with a mapping quality of zero are not used in variant calling. MAQ handles reads up to 128 bp in length; gapped alignment is only supported for paired-end and mate-pair reads.

3. **SHRiMP**: SHRiMP [90] is one of the few short-read aligners which support gapped alignment for single reads. It employs a fast *k*-mer hashing step [91] and a simple, very efficient vectorized implementation of the Smith-Waterman algorithm to find the mappings. Unlike

most of the other tools in this category, it has a lot of parameters that need to be adjusted based on the expected rate of polymorphism and this is one of the main reasons that it is not very popular.

4. **RMAP**: RMAP [92] is one of the earliest aligners for short reads. Features include support for base quality scores, best unique alignments for the reads and gapped alignment. Another tool like RMAP, but which works in a cloud environment is CloudBurst [93].

5. **ZOOM**: The approach taken by ZOOM [94] is similar to that of Eland but it allows use of spaced seeds [95]. It handles reads up to 224 bp in length, and has native support for paired-end mapping, quality scores, and gapped alignment. Another aligner which uses a similar approach to Eland with support for gapped alignments is SeqMap [96]

6. **MrsFast**: This aligner is designed specifically for reporting all hits. This is a useful feature for copy number variation (CNV) studies.

Some other popular tools include **cross_match** which is part of PHRAP [15], and has been modified to support short reads. Another aligner **RazerS** [97], uses q-gram filtration [91] to find the best hits and is based on the SeqAn library [98].

B.2 Indexing of Reference Genomes with Hash Tables

Some of the most popular aligners in this category are:

1. **SOAP**: SOAP [99] was the first published short read aligner. It supports paired-end mapping, adaptor trimming, gapped alignments and a SNP calling pipeline.

2. **NovoAlign**: NovoAlign [100] is definitely the most popular proprietary short-read aligner. It aligns reads with up to 8 mismatches per read read and supports gapped alignment on single and paired-end reads. It calibrates and uses base qualities at every step of the alignment for greater accuracy. It is also capable of $3'$ and $5'$ adaptor trimming for the Nimblegen Sequence Capture Arrays.

3. **MOSAIK**: MOSAIK [101] is a mutithreaded application that produces gapped alignments and assemblies. It also supports co-assembly or hybrid assemblies, where reads from multiple sequencing technologies can be mixed together.

4. **BFAST**: BFAST [102] finds the candidate alignment locations for each of the reads and then uses the Smith-Waterman algorithm to fully align them. A post-processor is then used to choose the best alignment.

5. **MOM**: MOM [103] maps reads with either a small number of mismatches in the entire read, or a small number of mismatches in the segment remaining after trimming bases from the $3'$ end or a single base from the $5'$ end.

6. **PASS**: PASS [104] creates an index of the genomic positions of "seed" words (typically 11 and 12 bases) as well as an index of the precomputed scores of short words (typically seven and eight bases) aligned against each other. After building the genomic index, the program scans every query sequence performing three steps: (1) it finds matching seed words in the genome; (2) for every match checks the precomputed alignment of the short flanking regions; (3) if passes step 2, then it performs an exact dynamic alignment of a narrow region around the match.

B.3 Indexing of Reference Genomes with BWT/Suffix Array

Representative aligners for this category include:

1. **SOAP2**: SOAP2 [105] is a versatile aligner that can support a wide range of read lengths, using a 2-way-BWT. It is capable of finding all hits for a read.

2. **BWA**: BWA [106] is another BWT-based aligner, and is similar to MAQ in features. It performs paired-end mapping, gapped alignments, and has an option to generate the output in SAM format [107].

3. **Bowtie**: Bowtie [108] is the fastest aligner for short reads and can handle reads up to 1024 bp in length. It does not work with ABI/SOLiD reads, and does not support gapped alignments, which severely limits its utility.

Vita

Aakrosh Ratan

Aakrosh Ratan is a Ph.D. candidate in Computer Science and Engineering at Penn State University. He received his B. Tech. (Bachelor of Technology) in Electrical Engineering from Indian Institute of Technology, Roorkee, India in 2003 and his S.M. (Master of Science) in Computer Science from National University of Singapore, Singapore in 2004. Following a brief stint as a R&D Engineer in the industry, he entered the Ph.D. program in Penn State in 2005 and shortly thereafter joined Webb Miller's lab at the Center of Comparative Genomics.