2014-12-01

The Bioluminescence Heterozygous Genome Assembler

Jared Calvin Price
Brigham Young University - Provo

The *Bioluminescence* Heterozygous Genome Assembler

Jared C. Price

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Mark J. Clement, Chair
Quinn O. Snell
William A. Barrett

Department of Computer Science

Brigham Young University

December 2014

ABSTRACT

The *Bioluminescence* Heterozygous Genome Assembler

Jared C. Price
Department of Computer Science, BYU
Master of Science

High-throughput DNA sequencing technologies are currently revolutionizing the fields of biology and medicine by elucidating the structure and function of the components of life. Modern DNA sequencing machines typically produce relatively short *reads* of DNA which are then assembled by software in an attempt to produce a representation of the entire genome. Due to the complex structure of all but the smallest genomes, especially the abundant presence of exact or almost exact repeats, all genome assemblers introduce errors into the final sequence and output a relatively large set of *contigs* instead of full-length chromosomes (a contig is a DNA sequence built from the overlaps between many reads). These problems are dramatically worse when homologous copies of the same chromosome differ substantially. Currently such genomes are usually avoided as assembly targets and, when they are not avoided, they generally produce assemblies of relatively low quality. An improved algorithm for the assembly of such data would dramatically improve our understanding of the genetics of a large class of organisms.

We present a unique algorithm for the assembly of diploid genomes which have a high degree of variation between homologous chromosomes. The approach uses coverage, graph patterns and machine-learning classification to identify haplotype-specific sequences in the input reads. It then uses these haplotype-specific markers to guide an improved assembly. We validate the approach with a large experiment that isolates and elucidates the effect of single nucleotide polymorphisms (SNPs) on genome assembly more clearly than any previous study. The experiment conclusively demonstrates that the *Bioluminescence* heterozygous genome assembler produces dramatically longer contigs with fewer haplotype-switch errors than competing algorithms under conditions of high heterozygosity.

Keywords: genome, genome assembly, polymorphic, polymorphism, heterozygous, haplotype, algorithm

ACKNOWLEDGMENTS

I would like to thank my wife and children for enduring the long hours worked and the committee chair for consistently supporting my efforts through numerous revisions and refinements. I would also like to thank each member of my committee and the many lab members and other scholars who helped to refine these ideas.

Table of Contents

List of Figures

List of Tables

Chapter 1

Problem Definition and Importance

The discovery of deoxyribonucleic acid (DNA) as the mechanism of heredity [1] is among the most significant discoveries in the history of science. It has already revolutionized our understanding of life and our efforts to sustain it, yet this revolution is clearly still in its infancy. In this chapter we briefly outline the improvements in human knowledge that are currently being realized through the study of DNA, and molecular biology in general, and the staggering promise this knowledge holds for dramatically improving human life with respect to medicine, nutrition and a host of other issues. We discuss at some length the central role algorithms are playing in this revolution and we introduce the whole genome assembly problem. In particular we introduce the whole genome assembly problem as it relates to species which have particularly high variation between homologous chromosomes and we discuss how an improved algorithm for this problem could promote better understanding of complex species and promote improvements in medicine, agriculture and more.

In the mind of a computer scientist, DNA can be usefully thought of as code, a remarkable kind of code expressing algorithms for the construction and regulation of proteins, and by extension, the construction and manipulation of life. Knowing how to program in this new language enables the engineering of life and its components. For example, it is now routine to harvest insulin from bacteria which have been engineered to produce it and we are fast approaching the routine production of transplant organs built from the patient's own DNA. As with more familiar algorithmic languages, slight variations in syntax can have dramatic semantic results. For example, the deletion or insertion of a single nucleotide in a

DNA molecule can dramatically change the sequence's meaning. For this reason, algorithms which infer the sequence of nucleotides in a genome must put an extraordinarily high value on accuracy.

Algorithms have become an essential part of the modern biologist's daily toolkit. The human genome is over 3 billion nucleotides long and encodes thousands of proteins which interact with each other, and with the body's incredibly complex environment, to produce the functions of life. Complexity is around every corner and the best tools we have in the modern world for managing this complexity are computers. For this reason, modern biology has necessarily become a computational science. Dr. Leroy Hood, 2012 winner of the National Medal of Science, put it this way "Biological information is divided into the digital information of the genome and the environmental cues that arise outside the genome. Integration of these types of information leads to the dynamic execution of instructions associated with the development of organisms and their physiological responses to their environments [13]."

We are in the midst of a profound revolution in human knowledge with respect to Biology and the mechanisms of life. This revolution is being enabled in no small part by advanced algorithms capable of elucidating complex biological structures and the even more complex interaction-networks between these components. The outcomes of this revolution are already profound and becoming more profound with each passing year. In the last few decades we have succeeded in determining, to a high degree of accuracy, the complete genome sequence of hundreds of important organisms, including the human genome [16, 31]. These blueprints have enabled us to study the components of life with an unprecedented level of completeness.

These developments have spawned the new field of computational biology in which computer experts build complex models to understand and predict the behavior of large biological networks. This field is rapidly ushering in the era of personalized medicine [10] in which millions of data points about each patient, including their personal genome sequence

and protein profile, can be leveraged to make more accurate diagnoses and prescribe more effective treatments.

In addition, improved understanding of the genetics of plants and micro-organisms provides an important avenue for the improvement of millions of lives by providing higher-nutrient foods to those in impoverished countries, helping to reduce greenhouse gases via bio-engineering, decreased drug development time, and a host of other issues.

Algorithms have a significant role to play in all of this. For example, Eugene Myers, a Computer Science PhD, dramatically accelerated the publication time of the human genome draft sequence by writing a genome assembly algorithm capable of inferring long contiguous sequences of DNA from much, *much*, smaller pieces that he stitched together *in silico* [22, 31]. Myers and his colleagues called their new approach the whole-genome shotgun technique [33], because it sequenced small, randomly-obtained, pieces of the genome and relied heavily on the accuracy of Myers' algorithm to reconstruct the original genomic sequence from which these small, randomly-obtained, pieces were drawn (see Figure 1.1).

When Myers first introduced the idea, many scientists believed the problem was simply too complex and could not be adequately solved. Myers' proved the critics wrong in a landmark publication which used his whole-genome shotgun technique to produce a draft sequence of the *Drosophila melanogaster* genome [22]. It was a watershed moment in the history of genomics and was enabled in large part by computer scientists.

In addition to genome assembly, algorithms have contributed to nearly every aspect of modern biology. Evolutionary relationships between organisms are now studied extensively using advanced algorithms for phylogeny estimation [11]. Complex protein interaction networks are now examined using detailed computational models of cells and their many components.

Biology has also given back to computer science by suggesting interesting ways to solve problems that mimic biological processes. One important example is the class of algorithms now known as genetic algorithms which subject populations of solutions to random mutation

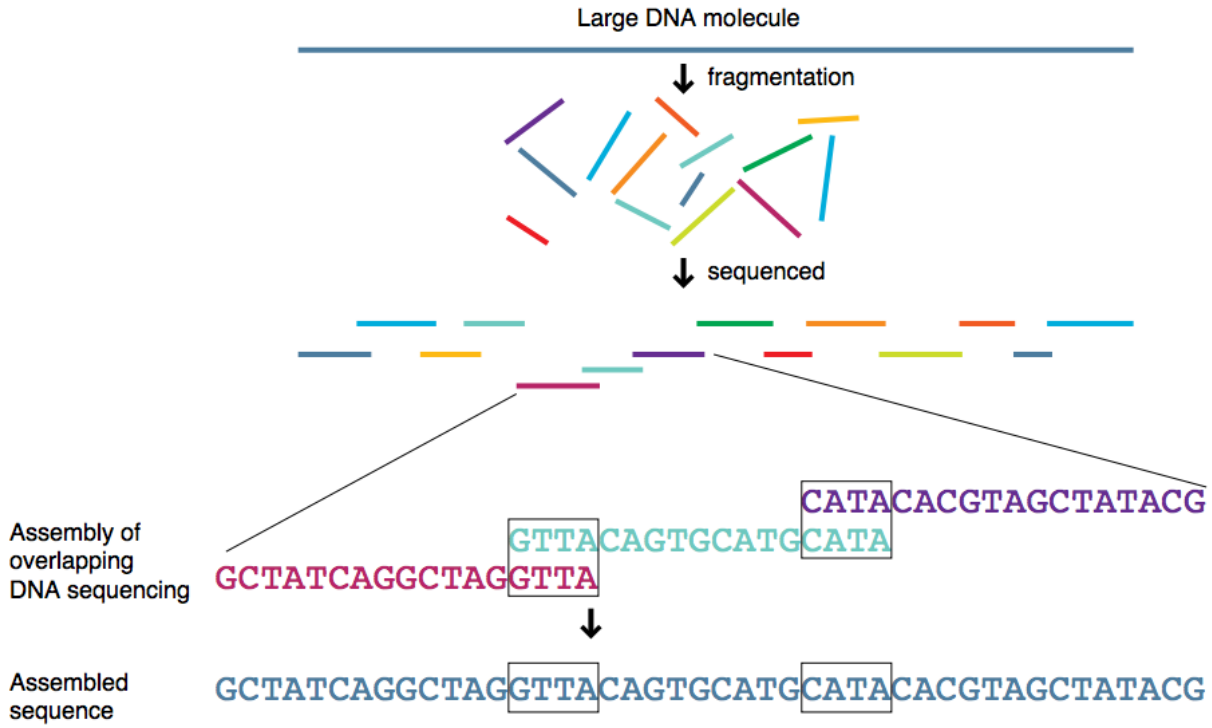

Figure 1.1: A simplified and very high-level overview of the whole-genome shotgun technique for genome assembly. Image reused from [25].

and "evolve" an optimized solution by subjecting the members of the population to survival pressures in the form of "fitness functions."

1.1 Genome assembly

In this work we will focus on the whole-genome assembly problem. More precisely, we will focus on the whole genome assembly problem as it relates to genomes that have large amounts of variation between pairs of homologous chromosomes. I will define the de novo reference genome assembly problem (DNRGAP) as

$$g : R \rightarrow C \tag{1.1}$$

where $R_e \in R$ is a large set of short DNA reads (usually less than 1000 nucleotides in length) sampled from a target genome of interest and $C_e \in C$ is a set of contigs (longer

sequences of DNA inferred from the information in R_e to be present in the target genome). DNRGAP is differentiated from the more general genome assembly problem by a couple of important properties. First, no reference genome for the organism can be used to solve a problem instance (hence de novo). Second, algorithms solving this problem aim to produce a single reference sequence when multiple homologous sequences are present in the genome. For example, if an organism has two homologous copies of a chromosome, algorithms for DNRGAP would attempt to output a single sequence representing both homologs.

It is common in assembly literature to use the term *contig* to refer to an assembled sequence that has no gaps (unknown bases) longer than a base or two, and to use the term *scaffold* to refer to a longer sequence with relatively large gaps of approximated size. In the definition of DNRGAP given above, an instance of C_e is intended to include both contigs and scaffolds. Finally, in a strict sense, the problem definition given above (because of the use of function notation) does not allow the use of non-deterministic assembly algorithms which cannot be guaranteed to map a specific input R_e to a specific C_e across multiple runs of the algorithm. For this reason I will explicitly clarify that the problem definition is not intended to preclude non-deterministic solutions.

The algorithm presented here introduces a novel approach for improving the contiguity and accuracy of algorithms for DNRGAP in the context of reads sampled from a highly polymorphic diploid individual. It is well understood that high polymorphism in the target genome is a source of assembly fragmentation and assembly error. However, good general-purpose solutions for this problem are still lacking as evidenced by the fact that nearly all published genomes involving highly-polymorphic sequence data include sections devoted to project-specific algorithms. Reference assemblies of polymorphic species are much more difficult to produce than their homozygous counterparts and are subject to a number of assembly errors. For example, two polymorphic *alleles* (different versions of a gene occupying the same locus) may be erroneously represented as *paralogs* (homologous sequences occupying different loci through gene duplication). Additionally, because algorithms for DNRGAP

attempt to produce a single representation of each locus, these representations can be mixtures of the two true haplotypes, producing contigs which are not actually present on either haplotype.

Solving DNRGAP would allow the genome assembly of a large class of organisms that are currently typically skipped over as targets for genome assembly simply because the difficulty of assembling the organisms is currently too great. It would also enhance our understanding of the true diversity between haplotypes in a wide array of species. It would give us a better understanding of the genes, proteins and regulatory mechanisms of a large class of highly polymorphic species which would in turn allow us to use this understanding to improve medicine, agriculture, and a host of other fields.

Chapter 2

Related Work

The body of related work can be organized into 4 classes, each of which is defined and discussed in its own subsection. Section 2.1 describes general-purpose algorithms for DNRGAP, and slight variations of the problem. Section 2.2 describes heterozygous genome assembly projects where the available tools were insufficient for producing a quality assembly. In each of these projects novel algorithms had to be developed to compensate for the weaknesses of the general-purpose solutions. Section 2.3 discusses algorithms for the inference of haplotypes when a reference genome is already available. Section 2.4 discusses genome assembly algorithms that specifically target polymorphic species.

2.1 General-purpose genome assembly algorithms

Algorithms for general-purpose genome assembly come in two major varieties: (1) overlap-layout-consensus (OLC) algorithms and (2) algorithms based on de Bruijn graphs.

The first step in the OLC approach is to calculate a pairwise alignment for each possible pair of reads. In the naive implementation, $\binom{n}{2}$ pairwise alignments must be computed (where n represents the total number of reads to be assembled). Given that large genome assembly projects routinely involve hundreds of millions of reads, an enormous number of pairwise alignments need to be computed (an assembly of 1 billion reads would require $5 * 10^{17}$ pairwise alignments). In practice, heuristics are used to eliminate large numbers of alignments. The most common heuristic involves checking each potential pair for a few short exact matches. Pairs which share no exact matches are not subjected to the full pairwise alignment algorithm.

Typically the pairwise alignments are used to construct an overlap graph where nodes represent reads and edges connect reads that overlap. After the pairwise alignments are calculated, and the overlap graph built, algorithms of this variety conceptually assign reads to genomic locations in a *layout* step and then construct a consensus genome sequence (typically a large set of contigs). Large assemblies performed using the OLC approach take days or weeks to compute. A thorough review of the computational complexity of various models of genome assembly is given in [20].

The de Bruijn graph approach is radically different [26]. In this formulation a read is represented by all of its substrings having a user-specified length k. These substrings are called k-mers and are organized into a de Bruijn graph. Each k-mer is represented as an edge in the graph with the source node representing the prefix of the k-mer sequence having length $(k-1)$ and the sink node representing the suffix of length $(k-1)$ [6]. This approach obviates the need for pairwise sequence alignments because exact matches are used and the overlaps between k-mers are inherently represented by the graph structure. A key insight is that this formulation (in the case of single-molecule circular genomes) allows the assembly problem to be framed as the problem of finding an Eulerian cycle as opposed to the much more computationally expensive task of finding a Hamiltonian cycle (as in the OLC case) [26]. This approach also has the advantage of representing repeats in a more natural way. The differences between the OLC approach and the de Bruijn graph approach are graphically depicted in Figure 2.1.

For both types of algorithms, heterozygous genomes present numerous challenges. For example, consider a particular location L in the genome and let L_a represent the sequence at that location on haplotype A and let L_b represent the sequence at that location on haplotype B. In order to produce an accurate haploid genome either the sequence L_a or the sequence L_b must be chosen. However, how is the assembler to know that L_a and L_b are not actually paralogs (copies of a gene occupying different locations in the same genome)? The assembler may believe L_a and L_b both need to be in the haploid genome thereby introducing error into

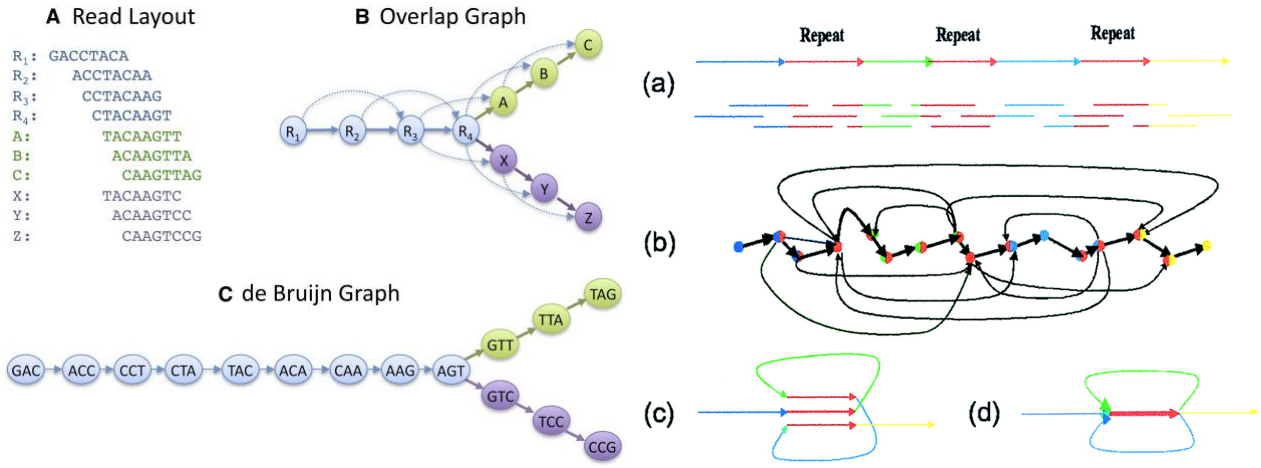

Figure 2.1: Differences between the OLC and de Bruijn graph approaches to the genome assembly problem. The image on the left-hand side is reused from [29] and shows how the overlap graph differs from the de Bruijn graph. The image on the right-hand side is reused from [26]. Section (a) of this image depicts part of a genome having a 3-copy repeat. It also shows reads sampled from this part of the genome. Section (b) shows the overlap graph built from the reads in section (a). Notice that the boundaries of each copy of the repeat are impossible to represent in the graph structure itself because traversing an edge adds much more than a single base. Sections (c) and (d) demonstrate that, because the de Bruijn graph has single-base resolution, the precise sequence that constitutes the repeat can be directly represented in the graph.

the final sequence. Now consider the case that the assembler realizes that L_a and L_b belong at the same location but produces a contig which is a composite of the two sequences rather than truly one variant or the other. In this case the assembler has produced contiguous sequence that doesn't actually occur on either haplotype! Furthermore, it is possible for a particular sequence of nucleotides to occur in one location on haplotype A and occur in a totally different location on haplotype B. Such sequence may assemble together into a single contig but appear to have two different, and conflicting, locations in the genome.

To date, there are no general-purpose, reliable, algorithms for producing high quality reference genome assemblies from reads that are sampled from 2 highly divergent haplotypes. This is underscored by the extensive use of project-specific pipelines for producing diploid assemblies. These approaches typically involve post-processing of a fragmented initial assembly. These studies will be discussed in more detail in section 2.2.

2.2 Project-specific algorithms for heterozygous genome assembly

To date, only a small number of highly heterozygous organisms have been assembled either into a reference sequence or into a complete genome assembly (an assembly where each homolog is represented by its own sequence). The inadequacy of algorithms for these tasks is underscored by the use of project-specific algorithms in each case [14, 17, 32, 35]. Two major approaches are used in these projects. The first approach is to allow for a greater degree of polymorphism at any genomic location in the building of contigs thereby allowing most polymorphic alleles to assemble together into single contigs, although these contigs are often a composite of the two haplotypes. This approach is only useful for relatively low rates of polymorphism and still requires manual curation to detect polymorphic loci that did not assemble together successfully. The other approach, and more successful one, is to purposely force the two divergent haplotypes to assemble separately as much as possible to produce a highly fragmented, but generally haplotype-coherent assembly. The fragmented contigs are then merged into a larger reference assembly by identifying pairs of sequences that appear to be homologous with respect to each other. The quality and contiguity of heterozygous genome assemblies, even when performed with significant manual intervention, is generally much lower than that which can be obtained using monoploid or homozygous data.

2.3 Haplotype assembly algorithms

The Haplotype assembly problem (HAP) refers to the problem of inferring haplotypes from a large set of DNA reads, when a reference genome for the organism is already available. This problem has been thoroughly studied as an optimization problem with most variants proved to be NP-hard [2, 19]. A polynomial time algorithm is known for the gapless case [2]. There is a rich set of innovative solutions for this problem that provide good performance in practice. For example, Bansal *et al.* describe a Markov Chain Monte Carlo (MCMC) method where the haplotype is represented as a point in very high dimensional space (a dimension

for each single nucleotide polymorphism (SNP)) [3]. The objective function minimizes the number of corrections that must be made to a set of reads to make them align perfectly with the solution haplotypes. Finding the actual optimal solution is non-polynomial, but MCMC allows sampling from the most probable regions of the haplotype space in polynomial time, generally producing a good approximation. Another innovative approach maps HAP to the familiar Max-2-SAT problem, thereby allowing the use of advanced SAT solvers for the haplotype assembly problem [21]. In general, these approaches are insufficient for two reasons. First, they typically target only SNPs and fail when a lot of insertion-deletion (indel) polymorphism is present. Second, these methods require that a reference assembly already be available.

2.4 Assembly algorithms for polymorphic species

There are relatively few genome assemblers which have been written specifically for the purpose of assembling data sampled from a single polymorphic individual although the last few years have seen a dramatic increase in polymorphic genome assembly research. One notable example is the *Hapsembler* algorithm which was developed as the PhD thesis of Nilgün Dönmez under well-known bioinformatics researcher Michael Brudno [7].

The *Hapsembler* algorithm [7, 8] is an OLC assembler which makes a unique contribution by formulating and constructing a *mate pair graph*. Conceptually, this structure is an overlap graph, however, it is a unique kind of overlap graph. Traditional overlap graphs have complete reads as their nodes and overlaps represented by edges. In Hapsembler's mate pair graph, the reads are not complete. From a long fragment it is possible to sequence 2 short reads, one from one end and one from the other end. Such reads are called paired end reads. Hapsembler's mate pair graph treats each paired-end read as a single long read and builds a structure analogous to a traditional overlap graph which shows the "overlaps" between these long reads and uses them to handle polymorphism more accurately than many other methods (as will be confirmed in the results section).

In addition to algorithms built specifically to deal with polymorphic genomes many general purpose assemblers such as Newbler (454 Life Sciences' proprietary assembler) and ALLPATHS [5] have features that are designed to improve their performance on heterozygous genomes. Newbler has a `-het` option which modifies the algorithm to better handle polymorphic genomes and ALLPATHS represents assemblies in graph form allowing researchers to more easily examine polymorphisms present in the input data.

Chapter 3

The *Bioluminescence* Heterozygous Genome Assembler

Thesis statement: The *Bioluminescence* Heterozygous Genome Assembler, which employs state-of-the-art machine learning techniques to aggressively discover haplotype-specific sequences during assembly, will produce longer contigs and less haplotype-switch errors than competing algorithms.

The *Bioluminescence* heterozygous genome assembler represents a very significant advance in the area of genome assembly algorithms for polymorphic species. The algorithm it uses is completely unique in the assembly literature and has the potential to allow for the robust assembly of species that are not amenable to traditional genome assembly approaches. To our knowledge, no other genome assembler uses state-of-the-art machine learning techniques in order to identify haplotype-specific sequences as a major component of the genome assembly pipeline. Bioluminescence uses this information to improve both contiguity and phasing of polymorphic genome assemblies.

The *Bioluminescence* heterozygous genome assembler is a k-mer centric genome assembler in the sense that the primary unit the assembler works on is the k-mer, not the read. However, unlike traditional k-mer based genome assemblers, Bioluminescence is not based on the de Bruijn graph model.

Instead, Bioluminescence attempts to solve the genome assembly problem in much the same way that a game player might attempt to win a game. Bioluminescence makes a series of discrete moves. Moves which have a higher probability of actually being part of the

correct assembly are made preferentially and moves continue to be made until no more moves meet the minimum threshold for quality.

Each "move" in the algorithm begins by selecting a k-mer. Care is taken to preferentially select k-mers which are more likely to produce good assembly results, as we shall discuss in more detail later. Each k-mer is the key into a hash table which can, at any time in the assembler's processing, fetch all of the intermediate sequences (reads or intermediate contigs) containing the chosen k-mer. A multiple-alignment of the sequences containing the k-mer is produced by using the position of the k-mer in the sequence as an "anchor" across which to align each sequence (see Figure 3.1).

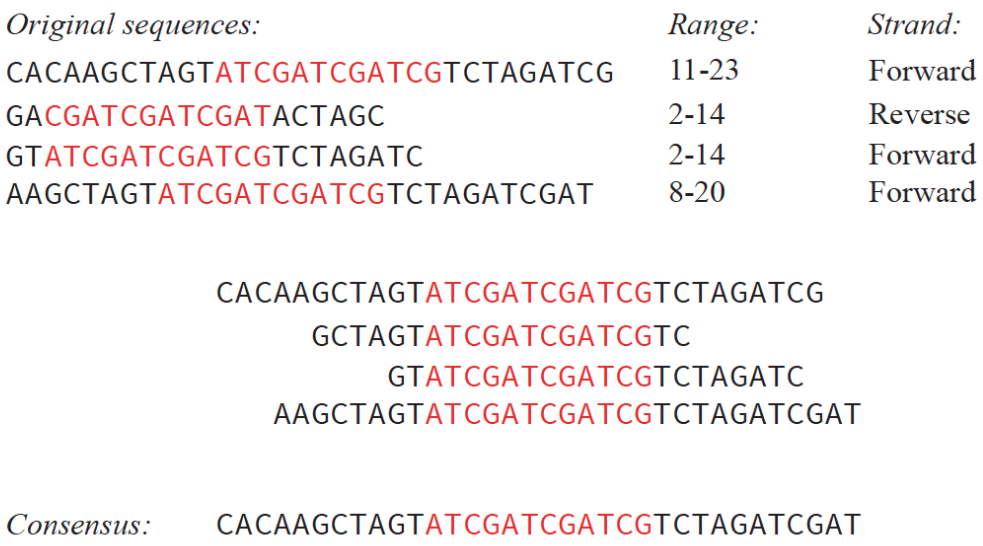

Figure 3.1: An example of how Bioluminescence builds a multiple alignment using a particular k-mer as an anchor. Bioluminescence first finds the location of the k-mer in each sequence, and the strand on which that k-mer is found. It then uses the knowledge about where the k-mer is in each sequence to perform an ultra-fast alignment by "normalizing" each sequence's position in the alignment such that the k-mer begins at the same location for every sequence in the alignment.

When such an alignment produces a *consistent* result, that is to say, an alignment in which all of the sequences agree (error handling is future work), the sequences are joined into a consensus sequence that itself can participate in later "moves" (also known as *joins*). The key to the Bioluminescence approach is that it makes moves that are likely to phase

haplotypes before it makes moves that might produce haplotype switch errors. By doing this, long, phased haplotypes are produced early on in the process and are preserved in later joins. As we will see in the validation chapter, this approach leads to better phasing in highly polymorphic genome assemblies.

The critical thing to understand at this point is that the choice of which k-mer to use for the anchor has dramatic effects on how the alignment should be treated when considering whether or not to perform a join. For example, if the anchor k-mer is haplotype-specific, meaning that it only occurs on one haplotype but not the other, then all of the reads in the alignment must have come from that haplotype, since they all contain the k-mer. This means that a join operation performed on such an alignment can *never* produce a haplotype switch error. It is this fact that Bioluminescence leverages to produce improved assemblies.

In contrast, if the anchor k-mer is present exactly once on both haplotypes, and in the same relative location on each, then a join operation performed on a consistent alignment may produce a haplotype switch error. If the anchor k-mer is present in multiple locations, of course, you then have the traditional repeat problem. Joins made on this class of k-mers have the potential to erroneously connect parts of the genome that are actually far apart from each other.

To gain deeper insight into how Bioluminescence works, it is important to understand the traditional model for counting repeats in genomes and how the Bioluminescence variant of this process is different. The traditional model for genome assembly has an implicit assumption that homologous sequences in diploid genomes are similar enough that they can be treated as if they are just a single molecule. For genomes which have very low polymorphism this assumption is reasonable to make and simplifies the assembly logic. Under this assumption a sequence is unique in the genome if it occurs on each haplotype exactly once. A 2-copy repeat would be a sequence that was found twice on haplotype A *and* twice and haplotype B and so on. There is also an implicit assumption that copies are in the same relative location on each homolog. For example, in the case of a 2-copy repeat there is an

implicit assumption that the 2 copies on haplotype A and the 2 copies on haplotype B are in similar positions on the 2 molecules.

This has profound implications for how edges in a contig graph are interpreted. Consider a very simple genome with a single 2-copy repeat that is much longer than any read in the input data. Let us refer to the first copy of the repeat as R_1 and the second copy of the repeat as R_2. Each copy of the repeat occurs in a different *context*. That is to say, the sequence that immediately surrounds R_1 is very likely to be different than the sequence that immediately surrounds R_2. In a traditional contig graph, this would most likely be represented as a single contig C which represents the repeated sequence. The depth of coverage of this contig would be likely to be roughly twice as much as for unique portions of the genome.

The contig C would be represented as a node in the graph and would have edges connecting it to both of the contexts (the sequences immediately surrounding the 2 copies of the repeat). Notice that under the assumption that the haplotypes are near identical the most likely interpretation of a bifurcation in the contig graph is that there is a 2 copy repeat. The assembler would be likely to search for a layout that is consistent with the edges which places that sequence in the genome twice. This is exactly the wrong thing to do if the bifurcation is actually caused by variation between haplotypes!

Variation between the 2 haplotypes causes a very similar graph pattern. Imagine 2 highly heterozygous haplotypes and consider a sequence that is present on both of them, and in the same location on both of them, but where the contexts are different in the 2 haplotypes. This would produce a graph structure very similar to the structure described in the previous paragraph but the coverage statistics would be markedly different. In this case, the analog of the 2-copy repeat is simply a sequence present on both haplotypes and it would be expected to be at a coverage typical for a sequence that is present just once on each haplotype. In contrast, the surrounding contexts, which are specific to each haplotype, would be expected to be at half the coverage of a single copy sequence present on both haplotypes. Strictly

graph-based approaches are likely to ignore clues in the coverage, and other clues, that might help the assembler understand that the bifurcation is caused by haplotype variation and not by a traditional 2-copy repeat. Bioluminescence aggressively identifies haplotype-specific sequences using a combination of coverage and contig graph patterns and is therefore able to understand which sequences are haplotype-specific to a much greater degree than traditional genome assembly algorithms. Figure 3.2 explains how copy-counting of sequences must be done very differently when haplotypes are highly heterozygous.

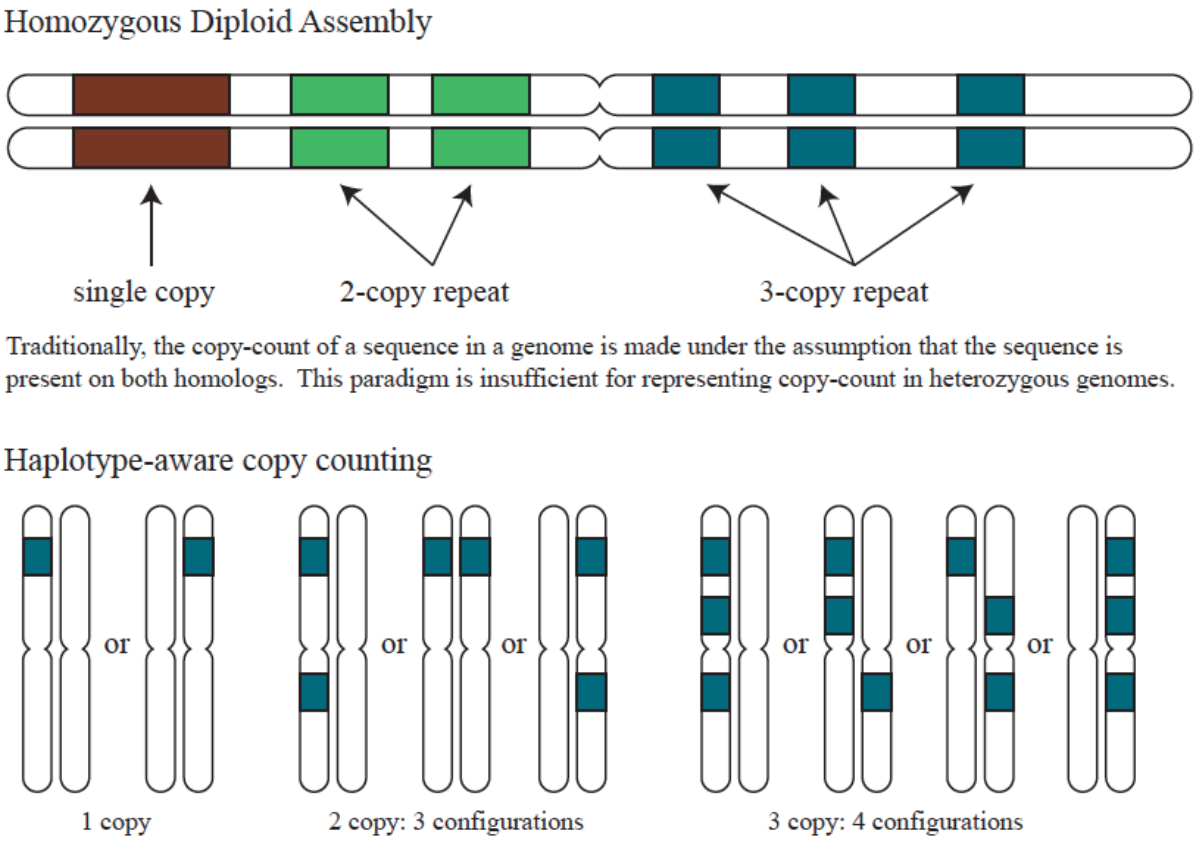

Figure 3.2: When a genome assembler assumes that homologous chromosomes are sufficiently similar that they can be treated as a single molecule it is likely to make mistakes when presented with highly heterozygous data. Counting how many times a sequence occurs in the genome is a much different task when the assembler assumes that the haplotypes have a high degree of variation from each other.

Bioluminescence identifies haplotype-specific markers in the assembly by using state-of-the-art machine learning classification techniques. First, it builds a feature table that contains a combination of coverage statistics and contig graph patterns. The contig graph patterns are drawn from a special kind of assembly performed using the Newbler software. In this step, Bioluminescence performs an ultra-stringent assembly of the data and explicitly instructs Newbler that there is no heterozygosity in the data. This forces Newbler to break contigs whenever there is significant variation.

The goal of this assembly is not to actually assemble the genome, rather the goal is to produce a contig graph which Bioluminescence can examine in order to learn from the structure of the graph. In this scenario it is desirable that the assembler break contigs at every significant point of variation and that contigs remain largely haplotype-coherent. This is why the assembler is parameterized as if no heterozygosity is present even though we expect heterozygosity.

Each row in the feature table reports information about a single k-mer. However, much of the data also requires knowing about the contig in which the k-mer is found in the ultra-stringent Newbler assembly. In order to allow this the Bioluminescence assembler contains a hash table which allows the software to quickly find the contig in which a particular k-mer is found in the Newbler assembly. The columns in the feature table are as follows.

1. The k-mer coverage.
2. The coverage of the contig in which the k-mer is found.
3. The length of the contig in which the k-mer is found.
4. Whether or not the contig in which the k-mer is found is one of the halves of a *perfect bubble* contig graph structure.
5. Whether or not the contig in which the k-mer is found is the haplotype-specific part of a *direct indel* contig graph structure.

6. Whether or not the contig in which the k-mer is found meets the *bifurcating neighbor* criterion.

Items 4-6 refer to specific contig graph patterns which have precisely defined semantics. The meaning of these structures will be discussed further in the next section. The goal of this section has been to introduce the reader to some of the core ideas in the algorithm not to provide implementation-level details. The next section will provide a more detailed discussion of the algorithm.

3.1 Detailed description of the algorithm

In this section we give a detailed description of the algorithm. The description of the algorithm given here attempts to be given in execution order, that is to say it attempts to present the algorithm in the order in which things occur when the program is executing. There may be slight deviations from this goal.

The algorithm begins by loading the unpaired reads into memory. Each read is then translated into a data structure which represents an *intermediate contig*. You can think of this data structure as the main substrate on which alignments, and subsequent join operations, occur. Each intermediate contig consists of the consensus DNA sequence for the contig and the alignment depth that has currently been observed at each position (as the algorithm progresses and intermediate contigs are joined together into longer contigs the alignment depth is updated).

The next step is to build an object which summarizes information about all of the k-mers in the unpaired read set. This structure consists of a hash table where a key is a particular k-mer and the value is an object summarizing information about that k-mer, specifically, how many times that k-mer was found in the unpaired reads and which k-mers are *neighbors* to the key k-mer in the sense that they are found together in one or more reads. This data structure also keeps track of how many reads the k-mers were found in together.

We then build a hash table designed to provide quick lookup from a particular k-mer to all of the intermediate contigs which contain that k-mer. As we will see shortly, this data structure is critical to the main assembly engine because for any particular k-mer the assembler needs to be able to find the set of intermediate contigs which contain that k-mer in constant time.

The next step is to train the classifier that will be used for determining whether a particular k-mer is haplotype-specific or not. For the purposes of this study the k-mer length is set to 31 and this k-mer length is used consistently for all k-mer-based genome assemblers in the study so that no assembler has an advantage over another.

As mentioned previously, the feature table used for classification incorporates both coverage statistics and contig graph patterns. In order to produce the feature table we need a contig graph we can learn from. This is produced in the following manner. First, we need a sequence to serve as haplotype A. Pretty much any real sequence will do, but preferably one that is relatively long so that it will provide many cases to learn from. A haplotype B sequence is then produced from haplotype A by randomly inserting SNPs into it. The current algorithm does this until it produces a haplotype B sequence that has, on average, 2 SNPs per 100 bases when compared to haplotype A.

Unpaired reads are then drawn randomly from the diploid genome to produce 40x coverage of each haplotype. These reads are then passed to a Newbler assembly that is specifically parameterized to require 100 percent overlaps and to expect that there is only 1 haplotype in the assembly. This forces the resultant contig graph to be quite fragmented under conditions of high heterozygosity. This is exactly what we want. Wherever boundaries between homozygous sequence and heterozygous sequence exist we want to force a bifurcation in the graph. We can then use these patterns, along with statistics about the depth of coverage of the contigs to learn about the k-mers.

Bioluminescence contains a data type that is capable of reading and understanding the Newbler contig graph and of searching it for particular patterns. This data type is used

to read in the Newbler contig graph and find 3 different types of patterns (1) the perfect bubble pattern (2) the indel pattern (3) the bifurcating neighbor pattern.

Consider a 100 base pair sequence on haplotype A and a different 100 base pair sequence, in the same location, on haplotype B. Consider also that the sequence flanking both of these differing sequences is the same. The tendency of the contig graph in this scenario would be to represent the 2 different sequences each in their own contig. We'll call the heterozygous sequence from haplotype A h_A and the heterozygous sequence from haplotype B h_B. Immediately flanking each of these sequences on the 3' end is a shared sequence we will call H_3. On the 5' end is a shared sequence we will call H_5. Notice that H_3 and H_5 are both going to need 2 edges, one to h_A and one to h_B. Any k-mers occurring in contig h_A or contig h_B would be given the value `true` for having the perfect bubble pattern if no other edges, in addition to the ones just described, connect to h_A or h_B. The intuition here is that because contigs h_A and h_B are very likely to be haplotype-specific we want the classifier to treat k-mers in these contigs differently than k-mers in contigs that are not likely to be haplotype-specific.

Now consider the same scenario, only this time instead of having h_B be a sequence of about the same length as h_A let's assume that h_B has simply been deleted from the genome. In this scenario both H_3 and H_5 have an edge to h_A but they also have an edge to each other. This creates a structure that looks kind of like a triangle and is what we mean by the "indel" pattern. In this case the contig h_A is likely to be haplotype specific and k-mers in this contig are marked `true` for having the indel pattern.

The final pattern is called *bifurcating neighbor*. A k-mer is said to have this property when the contig in which it is found in the Newbler assembly has the following properties.

1. Both ends of the contig (5' and 3') are adjacent to no more than 2 contig ends in the contig graph.
2. At least one of the 4 possible contig ends neighboring the contig participates in exactly 2 edges.

3. The contig is not either side of a perfect bubble structure.
4. The contig is not the haplotype-specific part of an indel structure.

Contigs meeting these criteria are more likely to be haplotype specific because this pattern can be produced when there is a boundary between sequence that is homozygous and sequence that is heterozygous. This occurs because the homozygous contig needs exactly 2 edges, one to the heterozygous sequence it is next to in haplotype A and one to the heterozygous sequence it is next to in haplotype B.

Bioluminescence then constructs a feature table for training. Each row in the feature table is a k-mer which was found in exactly one Newbler contig and which was also found in the reads sampled from the training diploid (these k-mers meet the minimum criteria necessary to produce values for each column in the table). For each such k-mer the following attributes are obtained.

1. The k-mer coverage in the input set of reads.
2. The coverage of the contig in which the k-mer is found.
3. The length of the contig in which the k-mer is found.
4. Whether or not the contig in which the k-mer is found is one of the sides of a *perfect bubble* contig graph structure.
5. Whether or not the contig in which the k-mer is found is the haplotype-specific part of an *indel* contig graph structure.
6. Whether or not the contig in which the k-mer is found meets the *bifurcating neighbor* criterion.

We are almost ready to train the model now. Bioluminescence contains a data structure which represents a diploid reference genome and its 31-mer spectrum on each haplotype. Using this structure Bioluminescence can know whether a k-mer is actually haplotype specific in the training genome or not. With all of the aforementioned features, and the class now

known, Bioluminescence uses the Weka [9] machine learning package to train a Random Forest classifier. The Random Forest classifier is used based on excellent classification accuracies in 10x cross validation within the training set (greater than 99% accuracy) and the results of an early unpublished pilot study related to this work.

We are now ready to classify the k-mers of the input data as either haplotype-specific or not haplotype-specific. In order to use the classifier, Bioluminescence follows steps analogous to what has just been described in order to produce a feature table with k-mers ready to be classified. Each one is then classified using the Random Forest classifier.

Now the assembly engine is ready to run. The assembly engine tries hard to assemble haplotype-specific sequence before it assembles other parts of the genome. The intuition behind this is that if these haplotype-specific sequences are built into long haplotypes first they will be preserved in later joins and produce a final assembly with less haplotype-switch errors than the competitors. As the validation section will prove, this is exactly what happens and the reason why Bioluminescence produces contigs that are more highly phased than the competing algorithms in this study.

At a high level this part of the assembly consists of selecting a k-mer to act as a *pivot* point, fetching all of the intermediate contigs that contain that k-mer, and then making a decision with regard to whether or not a set of intermediate contigs should be combined into a longer contig. Anchor k-mers which are likely to be haplotype-specific are chosen first. This is because any reads that contain such a k-mer must have been drawn from the haplotype that contains that k-mer, so joining them into a contig is a safe operation that will not produce haplotype switch errors.

Bioluminescence will order k-mers in such a way that k-mers that are more likely to be haplotype specific are considered as anchors before k-mers that are less likely to be haplotype-specific. More specifically, the k-mers are examined in the order of a priority queue. The priority queue is ordered as follows. First, k-mers are ordered according to the number of times they were observed in the input set of reads, with k-mers observed less often coming

first. The intuition is that k-mers at lower coverage are more likely to be haplotype-specific than k-mers at higher coverage. K-mer coverage is an integer value. It is just a count of how many times the k-mer occurred in the input set of reads. Because of this there are a *very* large number of ties. K-mers that occur at the same coverage are ordered in such a way that you always use k-mers (as anchors) that are classified as haplotype-specific before you consider those that are not classified as haplotype-specific. Bioluminescence continues this process of examining anchors and then either joining or not joining, based on whether the alignment is consistent, until its stopping criterion is met, after which it prints out the final collection of contigs.

Algorithm 1 Bioluminescence heterozygous genome assembler

```
1: Load the unpaired input reads into memory.
2: Transform each read into an intermediate contig data structure. Objects of this type will
   be used to construct alignments.
3: Build an object which summarizes properties of interest about the k-mer collection.
4: Build a hash table where a key is a particular k-mer in the assembly and the value is the
   set of intermediate contigs which contain that k-mer.
5: Build a machine learning model for classifying k-mers as haplotype specific.
6: Classify all of the k-mers in the assembly that have the requisite data for classification
7: Build a priority queue of k-mers where the least frequent k-mers are served first. Within
   a particular k-mer frequency serve all k-mers that are classified as haplotype-specific
   before serving the other k-mers at that frequency.
8: for all k-mers in queue (until stopping criterion met) do
9:    Produce a k-mer anchored alignment
10:   if the alignment is consistent then
11:      produce a longer contig from the component contigs and update the kmer-to-contig
         map
12:   else
13:      continue to the next k-mer
14:   end if
15: end for
16: Print final set of contigs.
```

Enhancing the core assembly algorithm to handle errors, building the paired-end module, and adding all of the other features that are still necessary to make Bioluminescence a general-purpose genome assembler is left for later work. At this stage in the algorithm's development it is intended as a vehicle to either prove or disprove the thesis statement. As we

will see in the validation section the thesis statement is dramatically validated. A simplified overview of the entire process is given in Listing 1.

Chapter 4

Validation of the Thesis Statement

This chapter presents the study that was conducted to validate the thesis statement. It has been written in a style suitable for independent publication. It is our opinion that this validation work contains the most powerful analysis yet published with regard to how genome assembly quality changes as a function of the SNP rate of the organism under study. In this work we perform a tightly controlled experiment in which changes in assembly quality can be directly attributed to changes in the SNP rate. We conclusively demonstrate that every assembler tested, except for Bioluminescence, exhibits *severe* reduction in genome assembly quality under high-SNP conditions even for the simplest known genomes.

The purpose of this research is to quantify the assembly quality achieved by a number of genome assembly algorithms when they are presented with the problem of assembling reads that have been sampled from a heterozygous diploid organism. It is well understood that such data are much harder to assemble than homozygous data [12, 14, 27, 30, 32, 35]. The papers cited here describe, among other things, many whole-genome reference assembly projects where the targeted organism exhibited a high degree of heterozygosity. The common theme in all of these projects is that the assembly of heterozygous data is particularly challenging. In many of these projects large and well-funded research teams had to resort to writing custom algorithms just to assemble their data and it remains unknown exactly how accurate these final assemblies are or how much better the contiguity of such assemblies could have been had the heterozygosity not been present. These studies suggest that it is important to understand

more about why heterozygous genome assemblies are often fragmented and inaccurate and to understand whether it is possible to improve algorithms for this type of data.

Despite the fact that it is well known that heterozygous diploid genomes are very challenging to assemble, often producing highly fragmented and/or error prone results, it is not yet well understood what specific strengths and weaknesses are exhibited by the popular genome assembly algorithms under these conditions. This lack of deep understanding about how SNP (single nucleotide polymorphism) rate, and other forms of heterozygosity, affect assembly quality can lead to a somewhat haphazard genome assembly approach which consists of assembling the data with an arbitrary selection of genome assemblers and selecting the assembly which produces the longest contigs. Such a strategy can lead to reference assemblies that are inaccurate, containing numerous haplotype-switch errors.

This work aims to achieve 2 goals. First, we aim to produce the most thorough analysis to date of how SNP rate affects genome assembly quality. Second, we wish to discover whether or not there is room for algorithmic improvement in this area by testing an approach, that is introduced with this work, which attempts to produce high quality reference assemblies even in the presence of highly heterozygous input data. We will reference these aims regularly throughout the manuscript, particularly in the section where we describe our experimental design decisions. When clarity calls for it, we may briefly repeat these aims in the body of the document, but for brevity we will sometimes refer to them simply as aim 1 and aim 2. These aims are highlighted and stated in a bit more detail in Table 4.1.

4.1 High-level overview of the experiment

In this section we present a very high-level overview of the entire experiment. Our goal in doing this is to provide at the very beginning of the manuscript a clear enough description of the entire experiment that the rest of the manuscript can be readily understood. In this section we will focus almost exclusively on a matter-of-fact description of *what* experiment

Aims of the experiment
1. To provide the most complete analysis to date, for the genome assemblers tested, of how genome assembly quality is affected by SNP rate when heterozygous diploid genomes are assembled by the whole-genome shotgun technique
2. To determine whether or not a new algorithm, the *Bioluminescence* haplotype-aware genome assembler, is capable of producing quantitatively improved genome assemblies in comparison to competing algorithms under the conditions of this study

Table 4.1: The aims of the experiment

was conducted and we will skip almost entirely a discussion about *why* we chose to design the experiment in this way. Immediately following this high-level overview of the experiment we will devote an entire section to a discussion of the design decisions.

The description of the experiment that is given in this section is not intended to be sufficiently detailed that it would allow full reproducibility of the experiment. For that level of detail the reader is referred to the Materials and Methods section.

In this experiment 60 diploid genomes, varying from each other only in SNP rate, are assembled. Each of these 60 genomes is assembled once by each of the tested algorithms, Bioluminescence, Hapsembler, Newbler, and Velvet, providing a total of 240 genome assemblies for analysis.

Each diploid genome consists of two molecules, a haplotype A molecule and a haplotype B molecule. Every one of the 60 diploid genomes shares the same haplotype A sequence, the NCBI reference sequence NC_021894.1, which is an extremely compact whole genome [23] that should be relatively easy for most assemblers to assemble with very high accuracy and very high contiguity from an input data set that does not also contain reads from a second, highly-related, molecule.

Sixty haplotype B sequences are used in the study, each one paired with the same haplotype A sequence to produce the 60 diploid sequences discussed in the previous paragraph. Each haplotype B sequence is the same length as haplotype A but is allowed to differ from

haplotype A by some number of single nucleotide polymorphisms while disallowing indel and rearrangement polymorphisms.

Each diploid genome can be uniquely identified within the set of 60 genomes by the percentage of positions at which the haplotype B sequence of that diploid genome differs from the haplotype A sequence. To be precise the 60 diploid genomes correspond in a one-to-one fashion to the SNP percentages in the set $\{0.0\%, 0.1\%, 0.2\%, 0.3\%, ..., 5.9\%\}$.

Each genome assembler (Bioluminescence, Hapsembler, Newbler, and Velvet) is given the exact same set of reads from which to try to assemble any particular genome. No errors are inserted into the reads and the data set includes both unpaired and paired-end reads. Each assembler is given 80x coverage of the genome (40x of each haplotype) in unpaired reads, each having a length of 150 base pairs. Each assembler also has access to two paired-end data sets. For both paired-end data sets the read length of each end is 50 base pairs. The first library has a mean insert size of 800 base pairs and a standard deviation of 80 base pairs. The second library has a mean insert size of 3000 base pairs and a standard deviation of 300 base pairs.

It should be clear at this point that the experimental design attempts to isolate 2 explanatory variables, the SNP rate and the genome assembly algorithm, from as many other variables as possible that also affect genome assembly quality such as repeat rate, input data set quality, rearrangement polymorphism etc.

At the highest conceptual level the study is interested in how these 2 explanatory variables affect a single response variable, the genome assembly quality. The tight control of other variables enables us to answer, with perhaps higher clarity than any previous study, questions such as "How does the SNP rate affect the genome assembly quality produced by algorithm X?", "Does algorithm X consistently produce better results under high SNP-rate conditions than algorithm Y?" and "Does algorithm X perform well under certain SNP rates but poorly under other SNP rates?"

In this study we will focus on 4 concrete response variables, each of which tells us something about genome assembly quality. These response variables are enumerated and highlighted in Table 4.2.

Response Variables
1. Contig N50
2. Max contig length
3. Total number of assembled bases
4. Percentage of the possible haplotype switch errors actually made

Table 4.2: The response variables analyzed in this study. Each variable is analyzed as a response to changes in the SNP percentage and the algorithm used for assembly.

4.2 Discussion of the experimental design

In this section we discuss in detail the design decisions we made. We discuss at some length the pros and cons of these decisions and provide the reasoning that was used to conclude that the design decisions are sound. This section is organized around specific design decisions.

4.2.1 The decision to use controlled genomes

The primary object of study in our experiment is how different genome assembly algorithms are affected by changes in the SNP rate (see Table 4.1). Once having settled on this object of study it would have been possible to attack the problem by first assembling a collection of real genomes that display a wide range of heterozygosity rates and to attempt to perform an assembly of each of these genomes with each of the tested genome assemblers.

Although such a study would be interesting, we believe it is clearly not the right choice for our aims. In this particular experiment we are attempting to attribute, with very high probability, any changes in our response variables to changes in our explanatory variables, SNP rate and algorithm. Most real genomes are incredibly complex and differ from each other across an enormous range of variables. For example, had we chosen to compile a collection of

previously-studied diploid genomes as the substrates for our study we would have had the following variables, each of which would have differed between each member of the collection.

1. Genome size
2. Genome complexity, as measured by a number of different variables such as repeat rate, extent of polymorphism, types of polymorphism etc.
3. The quality of the input data set, as measured by a number of different variables such as the quantity of reads obtained, the length of the reads obtained, the number of paired-end reads, the error rate in the reads, etc.

Some of these issues could have been mitigated to some extent, but not fully. For example, we could have decided that we would use previously-assembled heterozygous genomes as the substrates for our study and we could have attempted to normalize the quality of the input data sets by computationally producing comparable input data sets from each of these assemblies. Of course, in doing so, we introduce a new variable which is now the quality of the reads are dependent to some extent on the quality of those original assemblies.

Another major disadvantage to the approach described above is that there would be uncertainty in every measure. There would be some uncertainty as to the exact genome size, the exact SNP rate, the number of repeats, the types of repeats, the prevalence of indel/rearrangement polymorphism and more. In short, we believe that such a design would considerably reduce our ability to achieve the aims of the study.

4.2.2 The decision not to introduce errors into the reads

The decision to employ a design that uses tightly controlled, computationally-produced diploid genomes, necessitates the decision of using a computational process for sampling the reads from each genome. We decided the decision of not introducing errors into these reads was an appropriate design decision for this particular study for the following reasons. First, aim 2 involves testing, for the very first time, a new algorithm that is still under development.

It has been considered far more important to develop the primary concepts of the genome assembler than to focus on building yet another error-correction algorithm.

Of course, we could incorporate another group's error-correction algorithm as part of our genome assembly process, but doing so was not considered a high priority for the early stages of the algorithm's development.

A second possible advantage of the decision not to introduce errors into the reads is that doing so would add an additional variable that affects genome assembly quality. In other words, differences in assembly quality could be due in part to the relative effectiveness of the different algorithms at correcting the types of errors we introduced into the reads.

For these reasons we believe it is a sound design decision in this particular study to use error-free reads. With that said, a follow-up study will certainly have to be done that tests the ability of the Bioluminescence haplotype-aware assembler to correct sequencing errors as part of its pipeline.

4.2.3 The decision to use the 0.0% to 5.9% range

It was important to us to try and cover the entire range of published heterozygosity rates, and perhaps even a little higher, in order to cover almost all of the rates that will be found in real organisms. To our knowledge the *Ciona savignyi* assembly still has the highest reported heterozygosity rate of 4.6% [32]. We wanted to make sure the range went at least this high and decided to increase the top of the range to 5.9% since some organisms will certainly exhibit a higher rate than *Ciona savignyi*.

We also decided it was important to cover this range uniformly, something that the overall study design allows. For this reason we decided that our 60 genomes should map in a one-to-one way to the set $\{0.0\%, 0.1\%, 0.2\%, 0.3\%, ..., 5.9\%\}$. Having a uniform distribution over this range simplifies the task of talking about changes in a response variable that occur over a particular range of the explanatory variable because the same number of examples of the explanatory variable come from any 2 equidistant ranges of that variable.

4.2.4 The decision to focus on SNP-level polymorphism

Aim 1 of the study explicitly names the SNP rate as the explanatory variable in this study and not some other more general measure of heterozygosity. This language is intended to emphasize to the reader that this experiment explicitly excludes the analysis of how other forms of heterozygosity such as haplotype-specific rearrangements or differences in repeat content across the haplotypes affect assembly quality. These are equally important questions but are reserved for other studies because we aim to isolate our variable of interest from as many confounding variables as possible.

4.2.5 The decision to use NC_021894.1

It was important to us to use a real genome sequence, but we wanted that genome to be small for 2 reasons. First, since we are studying the affect of SNP rate, it is desirable to use a genome that doesn't have a large amount of repeats which would fragment the assembly regardless of SNP rate. In other words, it would be ideal if a genome assembler were capable of producing a near perfect assembly of the homozygous data so that we could clearly determine that any weaknesses in subsequent assemblies were directly attributable to the SNP rate. Secondly, we are testing a new algorithm and it was considered desirable to run it first on small genomes where the causes of any weakness in its approach could be more readily understood. These factors support the design decision to use the genome of *Carsonella ruddii* to play the role of haplotype A in each of the constructed diploid genomes.

Because the algorithm is under active development, even today, it was also necessary to have a large set of diploid genomes available for testing the algorithm as it progressed. Under the original design we had intended to do all algorithm testing on NC_021894.1 and use the related genome published in *Science* [24] as the substrate for the actual experiment. Due to time constraints this design decision had to be modified in favor of using NC_021894.1. This is mentioned because it is a potential source of bias, however, we believe that this decision is *highly* unlikely to have any appreciable effect on the conclusions of the study.

4.3 Benefits of the experiment

This study should give the clearest view to date into how different genome assemblers handle SNP-level variation in the underlying reads. Each assembler will have high-coverage, high-quality data from which to assemble and a small simple genome to reconstruct. By comparing the assemblies at low heterozygosity rates with the ones at high heterozygosity rates we should gain valuable insight into how each assembler handles such conditions and should be able to make more intelligent decisions about which assembler to use when presented with such data. We should also be able to draw conclusions about whether some of the techniques employed by the Bioluminescence haplotype-aware assembler provide quantitatively improved results in this scenario and whether these ideas are worth pursuing further.

4.4 Materials and methods

We begin this section with a brief discussion of the methods used by each of the genome assemblers tested. Bioluminescence is excluded from this section because it is described in detail in Chapter 3. This information will be of interest to most readers and will help place the results in their proper context. The second major subsection of this section provides the complete description of how the experiment was performed. There we provide a level of detail that allows for the experiment to be reproduced. We also give the precise definitions of each of the metrics we use in the results section.

4.5 The algorithms used for assembly

In this section we provide a brief overview of each of the algorithms tested. We will not provide an exhaustive explanation of each algorithm. Interested readers can find that information in the cited literature. The purpose of the explanations given here is to provide the reader with enough of an understanding of each of the algorithms to put the results of this study in their proper context. The algorithms are presented in the order they were released. This is

the most convenient ordering to use because genome assembly methods often build on one another.

4.5.1 Newbler

Newbler is a commercial genome assembler developed by *454 Life Sciences* primarily for the purpose of assembling data produced from their flagship sequencing machine *The GS FLX+ System*. Due to its proprietary nature its precise algorithm is unknown to the academic community at large although it is generally classified as being from the family of overlap-layout-consensus algorithms [18].

The algorithms placed in this category typically begin by building a traditional overlap graph or a highly-related data structure. In a traditional overlap graph each read becomes a node in the graph. When the sequences of 2 reads overlap with one another above a certain threshold an edge is placed in the graph to represent this overlap (See Figure 2.1).

Overlap graphs, and their relatives, are well represented in the genome assembly literature and have been used primarily for longer-read, lower-coverage, genome assemblies. The primary competitor to this class of algorithms is a class of algorithms based on de Bruijn graphs which we will discuss in more detail later.

One of the advantages of overlap-based assembly algorithms is that they can be easily used to express overlaps of varying length and quality. For example, R_1 may overlap with R_2 by 40 base pairs, and have 1 mismatch in the overlap, while R_3 overlaps with R_2 by 300 base pairs and has no mismatches in the alignment. These overlap differences can be easily represented in an overlap graph and can be used to give certain overlaps greater weight. As we will soon see, algorithms based on de Bruijn graphs represent the reads in a fundamentally different way.

Newbler performed very well in the Assemblathon 2 genome assembly competition where it exhibited "the highest levels of coverage and validity, and lowest values for multiplicity and parsimony among all competitive bird assemblies" [4].

4.5.2 Velvet

Velvet's genome assembly algorithm uses the classical de Bruijn graph approach that was pioneered by Pavel Pevzner. In this approach the reads are not directly represented in the graph. Instead they are broken into equal length fragments, known as k-mers and a graph is constructed using overlaps that all have length $k-1$.

This approach has several advantages. First, it scales better than the overlap-graph approach. In an overlap-graph you have to add a node for every read that you add to the assembly. This means that the graph size grows linearly with the number of reads, which in high coverage assemblies can be enormous.

In contrast, in the absence of errors, the de Bruijn graph has a hard limit on the number of nodes it will contain. Because there is a finite number of k-mers that actually exist in the genome there is also a finite number of nodes in the de Bruijn graph of any particular genome in the absence of read errors (which add erroneous k-mers to the graph). This issue of errors can of course be mitigated by removing low coverage k-mers that are likely to be errors.

Another advantage of the de Bruijn graph approach is that as you move from node to node across an edge you are moving at 1-base resolution. This is a nice property and simplifies the representation of repeats in assembly graphs.

The Velvet assembler has become a quite highly cited algorithm. As of September 17, 2014, Velvet [34] has been cited 2,974 times according to Google Scholar.

4.5.3 Hapsembler

Hapsembler [8] is an algorithm that was specifically designed to assemble highly polymorphic genomes. It was one of the very first assemblers to specifically tackle the problem of maintaining phase when 2 highly heterozygous haplotypes are being assembled together.

It is based on overlap-graph concepts, like Newbler, but it augments the traditional overlap graph by also constructing a *mate pair graph*. This mate pair graph is a pseudo-

overlap-graph which treats paired reads as if they were long single reads. Hapsembler uses this mate pair graph to exclude certain paths that would be available in the traditional overlap graph thereby excluding some paths that would cause haplotype-switch errors. This makes Hapsembler a particularly interesting algorithm for this study because it specifically targets the assembly of highly polymorphic input data, as does Bioluminescence.

4.6 The experiment

In this section we will describe in detail the experiment we conducted in order to validate the thesis statement. The experiment is a significant contribution in its own right. To our knowledge, it is the first experiment of its kind. Specifically it is the first published controlled experiment which increments SNP rates across a wide range of values and measures how genome assembly quality changes in response.

4.6.1 Constructing 60 genomes for assembly

We produced 60 diploid genomes to use for assembly, each consisting of a haplotype A sequence and a haplotype B sequence. Each diploid genome had the exact same haplotype A which was the NCBI (National Center for Biotechnology Information) reference sequence having accession number and version NC_021894.1.

For each of the 60 diploid genomes a unique haplotype B sequence was computationally produced by inserting single nucleotide polymorphisms into the haplotype A sequence. This was performed using custom software in the *Bioluminescence* library of bioinformatics tools. The algorithm works as follows. First the user selects the desired SNP probability in the final diploid sequence. The algorithm then calculates the smallest number of SNPs that need to be inserted to produce a SNP probability that is greater than or equal to the desired probability. The SNPs are inserted as follows. First a random single-nucleotide location in the genome is selected for mutation. If the position has already been previously selected and mutated the algorithm randomly selects a new location. Once a position has been found

that was not previously mutated either a transition event or a transversion event will occur. Because transition events are known to be much more probable than transversion events, the algorithm uses transitions 90% of the time. The algorithm limits itself to using the 4 canonical nucleotides A, C, G, and T. In other words, it doesn't insert ambiguous or unknown bases. In the case of a transversion the algorithm selects the 2 candidate bases at equal rates.

Using this technique, each of the 60 genomes for assembly was given a unique percentage of positions that contained SNPs. The set of SNP percentages is $\{0.0\%, 0.1\%, 0.2\%, 0.3\%, ..., 5.9\%\}$

4.6.2 Constructing the reads for assembly

From each of the 60 diploid genomes a set of reads to be used for assembly was constructed. Each complete read set consisted of 3 important subsets of reads. The entire set of reads used for each assembly is summarized in Table 4.3.

Paired	**Coverage**	**Errors**	**Mean insert length**	**Std. deviation**
no	80x (40x of each haplotype)	0	N/A	N/A
yes	4x (2x of each haplotype)	0	800	80
yes	4x (2x of each haplotype)	0	3000	300

Table 4.3: For each diploid genome a collection of reads to be used for assembly was produced. This table summarizes the reads produced for each genome. Each assembler used the exact same set of reads for assembly.

The reads were produced using version 0.9.5 of the MetaSim software [28]. The MetaSim software allows you to construct meta-genomes (collections of genomes) and to sample reads from them. In our case each meta-genome consists of 2 sub-genomes (the haplotypes).

4.6.3 Assembly parameters

In this section we report on the software versions and parameters that were used for assembly. The Newbler assemblies were performed using version 2.8 of the software. Newbler was

passed the `-het` option to ensure that the variant of the algorithm intended for heterozygous genomes was run. Paired-end reads were explicitly marked as paired-end reads so Newbler would not have to try and determine this on its own. Finally, Newbler was instructed to only print out contigs having length 151 base pairs or greater. This number was chosen because it is one base longer than the read length and was used with every assembler in order to provide a fair comparison.

Version 1.2.10 of the velvet software was used. Velvet was run with k-mer length 31. All k-mer based assemblies were run using the same length in order to ensure that no k-mer based method had an advantage simply because of the chosen k-mer length. The `velveth` command marked the paired-end reads, which had a mean insert size of 800, with `-shortPaired`. The other paired-end library was marked with `-shortPaired2`. The unpaired reads were marked with `-long` and all reads were marked with `-fasta`. The `velvetg` command used `-cov_cutoff auto` and `-read_trkg yes`. The `velvetg` command was also told the correct insert size and standard deviations for each paired end library and used the `-scaffolding yes` option.

Version 2.21 of the Hapsembler software was used. The Hapsembler assemblies were run using the `hapsemble` command. The reads were marked as having come from the Illumina platform. The genome size was estimated to be 160,000 base pairs. Because Hapsembler requires fastq reads, high quality values were given to all bases (Phred score = 40).

As already mentioned Bioluminescence was run using a k-mer size of 31. Finally, when a response variable could only be computed from an alignment these alignments were generated using version 3.23 of the MUMmer software [15]. The `nucmer` command was parameterized with `-maxmatch` and set the minimum length of a cluster of matches to 100.

4.7 Results and discussion

We will begin our examination of the results by looking at the effect of SNP rate, and algorithm choice, on the most ubiquitous assembly quality statistic, the contig N50. Although

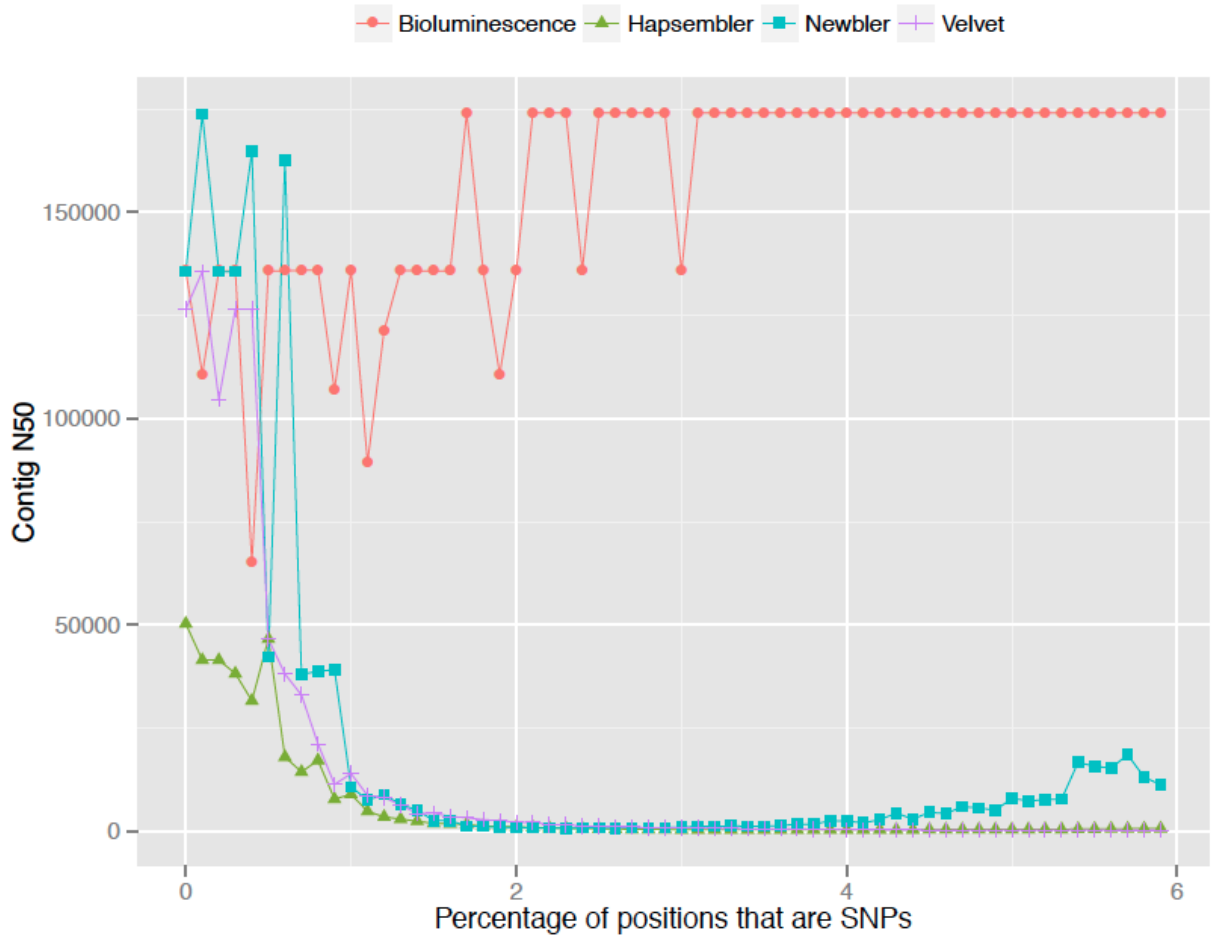

Figure 4.1: The contig N50 statistic calculated for each assembly performed in the study. Each point in the graph represents the N50 statistic calculated from a particular assembly.

the definition of the contig N50 statistic will be familiar to most readers, we repeat it here because there are a few minor variations on the precise mathematical meaning of the N50 statistic. The definition we use in this work is as follows. Given a set of contigs C the contig N50 for C is the longest contig in the set for which at least half of the total number of bases in the set are in contigs of that length or greater.

The N50 results are presented in Figure 4.1. The most striking characteristic of the N50 results is the degree to which contiguity degrades as the SNP rate increases. To get a bit of perspective on just how dramatic the decrease is in contig N50 consider the following. The contig N50 for the Newbler assembly on the 0.0% SNP rate genome is 53 times larger than

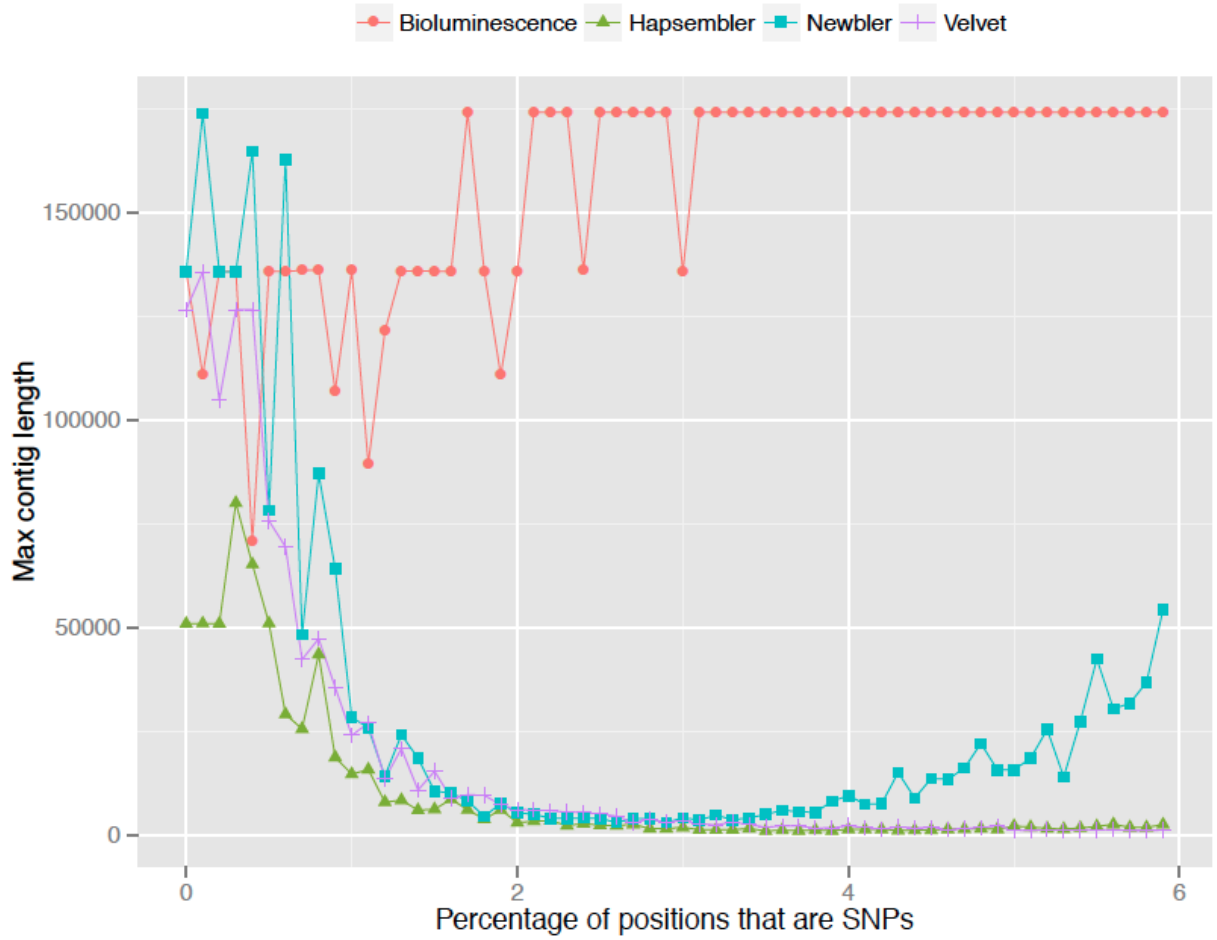

Figure 4.2: The max contig length statistic calculated for each assembly performed in the study. Each point in the graph represents the max contig length calculated from a particular assembly

the N50 Newbler achieves when the heterozygosity rate is 1.5%. This is a staggering result. Keep in mind that the heterozygosity presented in this paper is of the simplest possible variety. We have constrained it to SNP only variation and the genome being assembled is among the smallest and simplest to assemble in the world. This rate of degradation highlights a marked failure to assemble heterozygous genomes with any degree of quality. The numbers for Hapsembler and Velvet are similar in nature. Both algorithms show marked degradation in the contig N50.

The characteristics of the Bioluminescence algorithm are clearly unique among the set of tested algorithms. As the SNP rate increases, the Bioluminescence N50 does not degrade at all. In fact, it gets a bit larger, but it is consistently high across the vast majority of genomes. The total length of each genome is 174,014 base pairs, an N50 number which Bioluminescence either hits or nearly hits regularly.

The next response variable we will examine is the max contig length. It is likely that the pattern of these results will be very similar to the general pattern revealed by examining the N50 results. It is possible, however, for an assembly to have a small N50 and still have a relatively large max contig length. The results for max contig length are reported in Figure 4.2. As the figure immediately demonstrates, the pattern of the max contig length response variable closely follows the pattern shown in the N50 plot. All algorithms show a dramatic drop in the max contig length as heterozygosity increases except for the Bioluminescence algorithm.

Another interesting phenomenon related to the length of an assembly's contigs can occur in the presence of high levels of heterozygosity. It has been reported that for genomes having very high heterozygosity genome assemblers will sometimes produce 2 separate contigs for the same genomic location, one representing one haplotype and one representing the other haplotype. When this effect is prominent the total number of bases in the assembly should increase above the total number of bases in a monoploid representation of the genome.

In this study, we examine which assemblers produce this effect by reporting the total number of bases assembled for each assembly. These data are reported in Figure 4.3.

It is clear from these results that the behavior of different algorithms is quite a bit different with respect to whether or not they make the assembly much harder to interpret by including two contigs for single locations in the genome. Newbler and Hapsembler exhibit this behavior to the highest degree. This likely has something to do with the overlap-layout-consensus heritage of these 2 assemblers in contrast to the primarily k-mer based approaches of both Bioluminescence and Velvet.

Figure 4.3: The total number of bases in each assembly.

In a somewhat surprising result Velvet actually decreases the total number of bases as the heterozygosity rate increases over a certain range. The most likely interpretation of this result is that Velvet is not prone to producing 2 contigs for a single genomic location under the conditions of SNP-level variation, but that the contigs do get much smaller and in some cases become too small to be reported in the final results (only contigs 151 base pairs or greater are counted because this is 1 base longer than the read length), hence causing a decrease in the total number of bases reported for that assembly.

The total number of bases produced by Bioluminescence is remarkably consistent across all heterozygosity rates tested.

Figure 4.4: Shows the percentage of the total number of possible haplotype switch errors that were actually made in each assembly.

The final metric of assembly quality we will look at, and probably the single most important result, is the haplotype-switch error percentage. The haplotype-switch error percentage is calculated as follows. First, the total number of possible haplotype switch errors for a given assembly is calculated. This is calculated by mapping the contigs back to either one of the haplotypes, whichever it aligns to best. Then, if the contig covers 3 SNPs in the alignment we know that the maximum number of haplotype switch errors for that contig is 2. For a contig covering 5 SNPs the maximum number of haplotype switch errors is 4 and so on. The total number of haplotype switch errors for the entire assembly is then calculated and the actual number of haplotype-switch errors that are made is also calculated.

These numbers are then used to calculate the percentage of possible haplotype-switch errors that the assembler actually made for a particular assembly. If a contig has a base, at a SNP position, that is not the base on haplotype A or the base on haplotype B this counts as a haplotype switch error. Furthermore, because it is now impossible to say which haplotype the contig is on at that SNP position the next SNP position in the contig will also count as a switch error regardless of whether the base is on haplotype A or haplotype B.

This metric gives very valuable insight into how each assembler is actually producing its contigs. As we will see this metric demonstrates conclusively that Hapsembler and Bioluminescence far outperform their competitors in this metric, with Bioluminescence dominating the vast majority of the range of SNP rates. These results are given in Figure 4.4.

The 2 best algorithms with regard to the haplotype-switch error percentage are Hapsembler and Bioluminescence, with Bioluminescence being clearly the best algorithm for the vast majority of test cases. Hapsembler performs better than Bioluminescence across the range from 0.0% to 0.4% heterozygosity, but this is entirely to be expected because Bioluminescence currently lacks a paired-end module. Because of this, its benefits are only seen when the heterozygosity becomes high enough that multiple SNPs occur regularly within the read length. Until that point there is no reason to expect Bioluminescence to have any beneficial effect on this metric at all, which is essentially what we see.

For heterozygosity rates above 0.4% Bioluminescence demonstrates a *dramatic* improvement in haplotype-switch error rate in comparison to competing algorithms. This is the definitive proof of the thesis statement and the primary question this study was designed to answer. In short, the ideas Bioluminescence presents clearly show promise for improving the assembly of heterozygous organisms.

Chapter 5

Conclusion

Bioluminescence represents a very significant advance in the field of heterozygous genome assembly. Its primary contribution is that it introduces an entirely novel approach to heterozygous genome assembly that relies on coverage and advanced machine-learning techniques to identify haplotype-specific k-mers in an assembly. Bioluminescence then constructs alignments in which each constituent sequence of the alignment must contain a particular k-mer which is a marker for a particular haplotype. In doing so Bioluminescence is able to build haplotype-specific alignments and therefore, phased, haplotype-specific contigs. This is Bioluminescence's primary contribution and it is a completely new approach to the problem.

It is true that many diploid genomes have heterozygosity rates that are low enough that the techniques in Bioluminescence may provide little or no benefit, however, it is also demonstrably true that there are many organisms whose heterozygosity rates are sufficiently high that they could benefit from these methods. It is hard to estimate exactly what percentage of genomes have heterozygosity rates that are high enough to benefit from the techniques described here, but it is certain that the number is not trivial and it is likely underestimated by the currently published literature because homozygous genomes are preferentially chosen as assembly targets precisely because they are easier to assemble.

Via a very thorough experiment, we have convincingly shown that the ideas implemented by Bioluminescence are a significant advance for the field of genome assembly and

can be used to produce contigs from highly heterozygous input data that are generally longer and more accurately phased than the competing algorithms tested in this study.

References

[1] O. T. Avery, C. M. MacLeod, and M. McCarty, "Studies on the chemical nature of the substance inducing transformation of pneumococcal types: Induction of transformation by a desoxyribonucleic acid fraction isolated from pneumococcus type III," *The Journal of Experimental Medicine*, vol. 79, no. 2, pp. 137–158, 1944. [Online]. Available: http://jem.rupress.org/content/79/2/137.abstract

[2] V. Bafna, S. Istrail, G. Lancia, and R. Rizzi, "Polynomial and APX-hard cases of the individual haplotyping problem," *Theoretical Computer Science*, vol. 335, no. 1, pp. 109–125, 2005.

[3] V. Bansal, A. L. Halpern, N. Axelrod, and V. Bafna, "An MCMC algorithm for haplotype assembly from whole-genome sequence data," *Genome Research*, vol. 18, no. 8, pp. 1336–1346, 2008. [Online]. Available: http://genome.cshlp.org/content/18/8/1336.abstract

[4] K. Bradnam, J. Fass, A. Alexandrov, P. Baranay, M. Bechner, I. Birol, S. Boisvert, J. Chapman, G. Chapuis, R. Chikhi *et al.*, "Assemblathon 2: Evaluating de novo methods of genome assembly in three vertebrate species," *GigaScience*, vol. 2, no. 1, 2013. [Online]. Available: http://dx.doi.org/10.1186/2047-217X-2-10

[5] J. Butler, I. MacCallum, M. Kleber, I. A. Shlyakhter, M. K. Belmonte, E. S. Lander, C. Nusbaum, and D. B. Jaffe, "ALLPATHS: De novo assembly of whole-genome shotgun microreads," *Genome Research*, vol. 18, no. 5, pp. 810–820, 2008. [Online]. Available: http://genome.cshlp.org/content/18/5/810.abstract

[6] P. E. Compeau, P. A. Pevzner, and G. Tesler, "How to apply de Bruijn graphs to genome assembly," *Nature biotechnology*, vol. 29, no. 11, pp. 987–991, 2011.

[7] N. Dönmez, "Polymorphism and genome assembly," Ph.D. dissertation, University of Toronto, 2012.

[8] N. Dönmez and M. Brudno, "Hapsembler: An assembler for highly polymorphic genomes," in *Research in Computational Molecular Biology*, ser. Lecture Notes in Computer Science, V. Bafna and S. C. Sahinalp, Eds. Springer Berlin Heidelberg, 2011, vol. 6577, pp. 38–52. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20036-6_5

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[10] M. A. Hamburg and F. S. Collins, "The path to personalized medicine," *New England Journal of Medicine*, vol. 363, no. 4, pp. 301–304, 2010. [Online]. Available: http://www.nejm.org/doi/full/10.1056/NEJMp1006304

[11] M. Holder and P. O. Lewis, "Phylogeny estimation: Traditional and Bayesian approaches," *Nature reviews genetics*, vol. 4, no. 4, pp. 275–284, 2003.

[12] R. A. Holt, G. M. Subramanian, A. Halpern, G. G. Sutton, R. Charlab, D. R. Nusskern, P. Wincker, A. G. Clark, J. C. Ribeiro, R. Wides *et al.*, "The genome sequence of the malaria mosquito Anopheles gambiae," *Science*, vol. 298, no. 5591, pp. 129–149, 2002. [Online]. Available: http://www.sciencemag.org/content/298/5591/129.abstract

[13] L. Hood, J. R. Heath, M. E. Phelps, and B. Lin, "Systems biology and new technologies enable predictive and preventative medicine," *Science*, vol. 306, no. 5696, pp. 640–643, 2004. [Online]. Available: http://www.sciencemag.org/content/306/5696/640.abstract

[14] T. Jones, N. A. Federspiel, H. Chibana, J. Dungan, S. Kalman, B. B. Magee, G. Newport, Y. R. Thorstenson, N. Agabian, P. T. Magee *et al.*, "The diploid genome sequence of Candida albicans," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 19, pp. 7329–7334, 2004. [Online]. Available: http://www.pnas.org/content/101/19/7329.abstract

[15] S. Kurtz, A. Phillippy, A. Delcher, M. Smoot, M. Shumway, C. Antonescu, and S. Salzberg, "Versatile and open software for comparing large genomes," *Genome Biology*, vol. 5, no. 2, p. R12, 2004. [Online]. Available: http://genomebiology.com/2004/5/2/R12

[16] E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, K. Dewar, M. Doyle, W. FitzHugh *et al.*, "Initial sequencing and analysis of the human genome," *Nature*, vol. 409, no. 6822, pp. 860–921, 2001.

[17] S. Levy, G. Sutton, P. C. Ng, L. Feuk, A. L. Halpern, B. P. Walenz, N. Axelrod, J. Huang, E. F. Kirkness, G. Denisov *et al.*, "The diploid genome sequence of an individual human," *PLoS Biology*, vol. 5, no. 10, p. e254, 2007.

[18] Z. Li, Y. Chen, D. Mu, J. Yuan, Y. Shi, H. Zhang, J. Gan, N. Li, X. Hu, B. Liu *et al.*, "Comparison of the two major classes of assembly algorithms: Overlap-layout-consensus

and de-bruijn-graph," *Briefings in Functional Genomics*, vol. 11, no. 1, pp. 25–37, 2012. [Online]. Available: http://bfgp.oxfordjournals.org/content/11/1/25.abstract

[19] R. Lippert, R. Schwartz, G. Lancia, and S. Istrail, "Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem," *Briefings in Bioinformatics*, vol. 3, no. 1, pp. 23–31, 2002. [Online]. Available: http://bib.oxfordjournals.org/content/3/1/23.abstract

[20] P. Medvedev, K. Georgiou, G. Myers, and M. Brudno, "Computability of models for sequence assembly," in *Algorithms in Bioinformatics*. Springer, 2007, pp. 289–301.

[21] S. R. Mousavi, M. Mirabolghasemi, N. Bargesteh, and M. Talebi, "Effective haplotype assembly via maximum boolean satisfiability," *Biochemical and Biophysical Research Communications*, vol. 404, no. 2, pp. 593 – 598, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0006291X10022266

[22] E. W. Myers, G. G. Sutton, A. L. Delcher, I. M. Dew, D. P. Fasulo, M. J. Flanigan, S. A. Kravitz, C. M. Mobarry, K. H. J. Reinert, K. A. Remington *et al.*, "A whole-genome assembly of Drosophila," *Science*, vol. 287, no. 5461, pp. 2196–2204, 2000. [Online]. Available: http://www.sciencemag.org/content/287/5461/2196.abstract

[23] A. Nakabachi, R. Ueoka, K. Oshima, R. Teta, A. Mangoni, M. Gurgui, N. J. Oldham, G. van Echten-Deckert, K. Okamura, K. Yamamoto *et al.*, "Defensive bacteriome symbiont with a drastically reduced genome," *Current Biology*, vol. 23, no. 15, pp. 1478 – 1484, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0960982213007525

[24] A. Nakabachi, A. Yamashita, H. Toh, H. Ishikawa, H. E. Dunbar, N. A. Moran, and M. Hattori, "The 160-kilobase genome of the bacterial endosymbiont Carsonella," *Science*, vol. 314, no. 5797, p. 267, 2006. [Online]. Available: http://www.sciencemag.org/content/314/5797/267.abstract

[25] National Human Genome Research Institute, "Shotgun sequencing," in *Talking Glossary of Genetic Terms*. [Online]. Available: http://www.genome.gov/Glossary/resources/shotgun_sequencing.pdf

[26] P. A. Pevzner, H. Tang, and M. S. Waterman, "An Eulerian path approach to DNA fragment assembly," *Proceedings of the National Academy of Sciences*, vol. 98, no. 17, pp. 9748–9753, 2001, Copyright 2001 National Academy of Sciences, USA. [Online]. Available: http://www.pnas.org/content/98/17/9748.abstract

[27] Potato Genome Sequencing Consortium, "Genome sequence and analysis of the tuber crop potato," *Nature*, vol. 475, no. 7355, pp. 189–195, 2011.

[28] D. C. Richter, F. Ott, A. F. Auch, R. Schmid, and D. H. Huson, "MetaSim—A sequencing simulator for genomics and metagenomics," *PLoS ONE*, vol. 3, no. 10, p. e3373, 2008.

[29] M. C. Schatz, A. L. Delcher, and S. L. Salzberg, "Assembly of large genomes using second-generation sequencing," *Genome Research*, vol. 20, no. 9, pp. 1165–1173, 2010. [Online]. Available: http://genome.cshlp.org/content/20/9/1165.abstract

[30] R. Velasco, A. Zharkikh, M. Troggio, D. A. Cartwright, A. Cestaro, D. Pruss, M. Pindo, L. M. FitzGerald, S. Vezzulli, J. Reid *et al.*, "A high quality draft consensus sequence of the genome of a heterozygous grapevine variety," *PLoS ONE*, vol. 2, no. 12, p. e1326, 12 2007. [Online]. Available: http://dx.plos.org/10.1371%2Fjournal.pone.0001326

[31] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt *et al.*, "The sequence of the human genome," *Science*, vol. 291, no. 5507, pp. 1304–1351, 2001. [Online]. Available: http://www.sciencemag.org/content/291/5507/1304.abstract

[32] J. P. Vinson, D. B. Jaffe, K. O'Neill, E. K. Karlsson, N. Stange-Thomann, S. Anderson, J. P. Mesirov, N. Satoh, Y. Satou, C. Nusbaum *et al.*, "Assembly of polymorphic genomes: Algorithms and application to Ciona savignyi," *Genome Research*, vol. 15, no. 8, pp. 1127–1135, 2005. [Online]. Available: http://genome.cshlp.org/content/15/8/1127.abstract

[33] J. L. Weber and E. W. Myers, "Human whole-genome shotgun sequencing," *Genome Research*, vol. 7, no. 5, pp. 401–409, 1997. [Online]. Available: http://genome.cshlp.org/content/7/5/401.short

[34] D. R. Zerbino and E. Birney, "Velvet: Algorithms for de novo short read assembly using de Bruijn graphs," *Genome Research*, vol. 18, no. 5, pp. 821–829, 2008. [Online]. Available: http://genome.cshlp.org/content/18/5/821.abstract

[35] A. Zharkikh, M. Troggio, D. Pruss, A. Cestaro, G. Eldrdge, M. Pindo, J. T. Mitchell, S. Vezzulli, S. Bhatnagar, P. Fontana *et al.*, "Sequencing and assembly of highly heterozygous genome of Vitis vinifera L. cv Pinot Noir: Problems and solutions," *Journal of Biotechnology*, vol. 136, no. 12, pp. 38 – 43, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168165608001880