2013-06-25

Inverted Sequence Identification in Diploid Genomic Scaffold Assembly via Weighted MAX-CUT Reduction

Paul Mark Bodily
Brigham Young University - Provo

Inverted Sequence Identification in Diploid Genomic Scaffold Assembly via Weighted MAX-CUT Reduction

Paul M. Bodily

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

Master of Science

Department of Computer Science

Brigham Young University

June 2013

ABSTRACT

Inverted Sequence Identification in Diploid Genomic Scaffold Assembly
via Weighted MAX-CUT Reduction

Paul M. Bodily
Department of Computer Science, BYU
Master of Science

Virtually all genome assemblers to date are designed for use with data from haploid or homozygous diploid genomes. Their use on heterozygous genomic datasets generally results in highly-fragmented, error-prone assemblies, owing to the violation of assumptions during both the contigging and scaffolding phases. Of the two phases, scaffolding is more particularly impacted and algorithms to facilitate the scaffolding of heterozygous data are lacking. We present a stand-alone scaffolding algorithm, ScaffoldScaffolder, designed specifically for scaffolding diploid genomes.

A fundamental step in the scaffolding phase is the assignment of sequence orientations to contigs within scaffolds. Deciding such an assignment in the presence of ambiguous evidence is what is termed the *contig orientation problem*. We define this problem using bidirected graph theory and show that it is equivalent to the weighted MAX-CUT problem. We present a greedy heuristic solution which we comparatively assess with other solutions to the contig orientation problem, including an advanced MAX-CUT heuristic. We illustrate how a solution to this problem provides a simple means of simultaneously identifying inverted haplotypes, which are uniquely found in diploid genomes and which have been shown to be involved in the genetic mechanisms of several diseases. Ultimately our findings show that due to the inherent biases in the underlying biological model, a greedy heuristic algorithm performs very well in practice, retaining a higher total percent of edge weight than a branch-and-bound semidefinite programming heuristic.

This application exemplifies how existing graph theory algorithms can be applied in the development of new algorithms for more accurate assembly of heterozygous diploid genomes.

Keywords: Genome Assembly, Scaffolding, Weighted MAX-CUT Algorithms, Bidirected Graph Theory

ACKNOWLEDGMENTS

Thanks go my advisor, Dr. Mark Clement, without whose patient guidance and support this research would not have been possible. Thanks also go to the following other students, graduate and undergraduate, in the Computational Sciences Laboratory for their contributions of time, critical feedback, and support: Jared Price, Stanley Fujimoto, Nozomu Okuda, and Cole Lyman. I would like to acknowledge the time, effort, and support given by my committee which helped greatly to improve this thesis. I would also like to thank my wife, Courtney, whose constant patience, support, and encouragement contributed monumentally to the completion of this thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

An Introduction to Heterozygous Genome Assembly

1.1 Motivation

Deoxyribonucleic Acid (DNA) molecules are the building blocks of all forms of life on this planet, from viruses and bacteria to human beings. The unique combination of adenine (A), cytosine (C), guanine (G), and thymine (T) bases to form a DNA sequence is what ultimately engenders diversity between species and between individuals. Genetic variation is the root cause for numerous diseases or predispositions to life-threatening diseases such as cancer, heart disease, and HIV. Genetic variation in plants is the basis for variability in crop yields, nutritional value, and flavor. The future of scientific study in these areas depends heavily on the ability to study and characterize genetic variation.

Despite the direct bearing that genetics has on each of these instances, the specific genetic variations at play are not well-characterized, their effects are not well-understood, and the ability to scientifically study them is limited. This is due to the relatively sparse amount of accurately assembled data that is available. The shortage derives in large part from the cost-prohibitive and somewhat primitive nature of the technology and software used to obtain and analyze genetic data. The first human genome was assembled nearly a decade ago and cost close to 2.7 billion dollars[1]. Though DNA sequencing costs have decreased significantly, the time and manual effort required to produce finished genome sequences are still very restrictive. Despite global efforts to collect and sequence any and all forms of life, only about 1,200 organisms have been sequenced and are available in NCBI's public database—most at a preliminary level—hardly enough to begin

[1]`http://www.genome.gov/11006943`

to adequately characterize the patterns responsible for genetic variations of interest[2]. In order to deduce and characterize the effects of genetic variation, we need a larger number of high-quality sequenced genomes, implying the need for technology and software to produce them.

Genome assembly projects to date have aimed at generating a single reference sequence to represent a given species. However, the genomes of nearly all large species actually consist of two or more variated copies of such a reference. Because of the increased complexity involved in assembling two similar but different copies, most sequencing projects preliminarily require inbreeding of a target specimen before data is collected. This inbreeding step aims to reduce the variation and produce an organism with DNA that is more easily assembled. However, this preparatory work is both time- and cost-intensive and in some cases has raised ethical concerns. Often this homogenizing step is impossible because it yields unviable progeny. In all cases the unique variation between copies is lost, and with it, a great deal of critical data. There is a need for tools capable of directly assembling and analyzing genomes with more than one copy.

1.2 Background

The DNA of all plant and animal genomes is divided among a number of *chromosomes*. The number of chromosomes or *haploid number* is a defining characteristic for a given species, and the ordering and content of chromosomes is predominantly the same between individuals of the same species. Independent of the haploid number for a species is the *ploidy* of a species, a number which represents the number of copies of the entire set of chromosomes in the cell. Most animals are *diploid* organisms, meaning that their cells contain two copies of each chromosome, one deriving from each of two parents. There are slight differences between the copies (each unique copy is called a *haplotype*) which interact to make the offspring unique from either parent. Matching regions (i.e. sequences in the same position) on each copy are termed *homologous* regions or *homologs*. Two homologous sequences that are the same are called *homozygous*; the difference between homologs is expressed in terms of *heterozygosity*. When we speak of sequencing the DNA of a species (or

[2] `http://www.ncbi.nlm.nih.gov/About/tools/restable_mol.html`

simply "sequencing a species"), we are generally referring to finding an "average" sequence, where the nucleotide base at any chromosomal location is a function of the most common base among individuals of the species.

Technology has thus far been only moderately successful at solving the problem of genomic sequencing. The most prominent methods require large-scale replication of genetic material which is then broken through sonication into an amalgam of short fragments of various sizes (called *reads*). From this mixture are extracted sequences suitable to the capacity of sequencing machines. The most cost-effective machines are capable of sequencing reads of approximately 100 bases while maintaining reasonably low error rates. Reads as long as several hundred base pairs can be sequenced at a much higher price and with somewhat higher error rates. In any case, the sequencers are unable to sequence anything that even begins to approximate the size of an entire chromosome which, for example in a raspberry, averages lengths of several million base pairs. The algorithmic challenge is to reassemble the full-length chromosomes from short DNA reads.

This reconstruction is generally broken into two phases: the overlapping of reads to form consensus contigs and the scaffolding of contigs to form chromosomes.

1.2.1 Contigging

In the initial phase of the *assembly* process we seek to use the reads to form long contiguous sequences of known bases. This is accomplished by combining overlapping reads to produce longer consensus sequences called *contigs* (see Figure 1.1a). If all read-length genomic sequences were unique, we could continue this process until we reconstructed the original chromosomal sequence in its entirety. However, due to the presence of repetitive regions throughout a genome, reads will exist which support multiple paths of reconstruction (see Figure 1.1b).The ambiguity of this result is often modeled as a graph where the nodes are the unambiguous consensus contig sequences produced from combining overlapped reads and the edges are possible ways in which these contigs could be sequentially combined (see Figure 1.1c). Often the number of contigs can outnumber the actual number of chromosomes by as much as a factor of 10^3. The graph will often be missing

```
Read 1:      ACCCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGC
Read 2:       CCCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCA
Read 3:        CCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAG
Read 4:         CGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGA
Read 5:          GGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAA
Read 6:           GCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAAG
Read 7:            CGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAAGC
...

Consensus:   ACCCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAAGC...
```

(a)

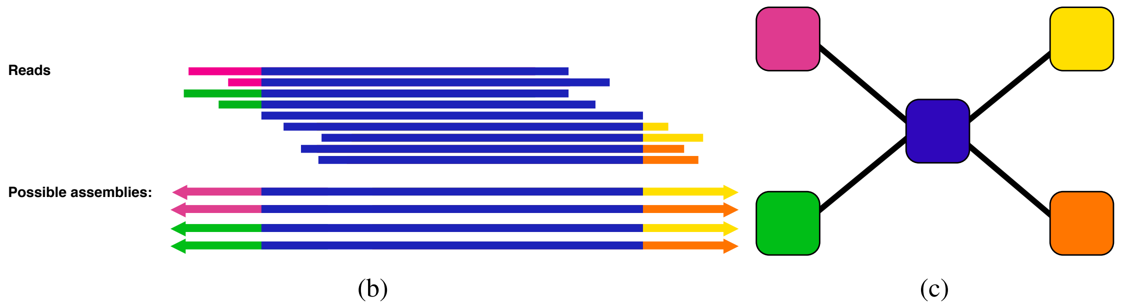


Figure 1.1: (a) Short reads whose sequences overlap are overlaid such that their consensus is a reconstruction of the original sequence from which the reads are taken. (b) Repetitive regions whose length exceeds that of sequenced reads create different possible reconstructed consensus paths. (c) The different reconstructions can be modeled as a graph where unambiguous consensus sequences are collapsed into nodes and evidence for the different paths are represented as edges.

nodes or edges due to insufficient sampling of certain areas of the genome or because of erroneous contigs produced from errors during the read-sequencing phase.

1.2.2 Scaffolding

Scaffolding is the step in the assembly process where additional information is leveraged to infer the relative distance and orientation of contigs. Two distinct approaches are commonly used in the scaffolding step. The first approach attempts to infer the distance and orientation of contigs by using the known sequence of a closely related genome. We refer to the degree of genetic similarity in gene-order between different species as *synteny*. To the extent that the genomes of two species are syntenic, this approach can be moderately successful. However, no species has the same genome as that of another species which inherently limits this approach to scaffolding.

The second approach to scaffolding makes use of *paired-end data*. Paired-end data consists of pairs of short reads whose distance and orientation is known from the technique used to sequence them. Due to the read-size constraints mentioned above, the paired-reads are the same length or

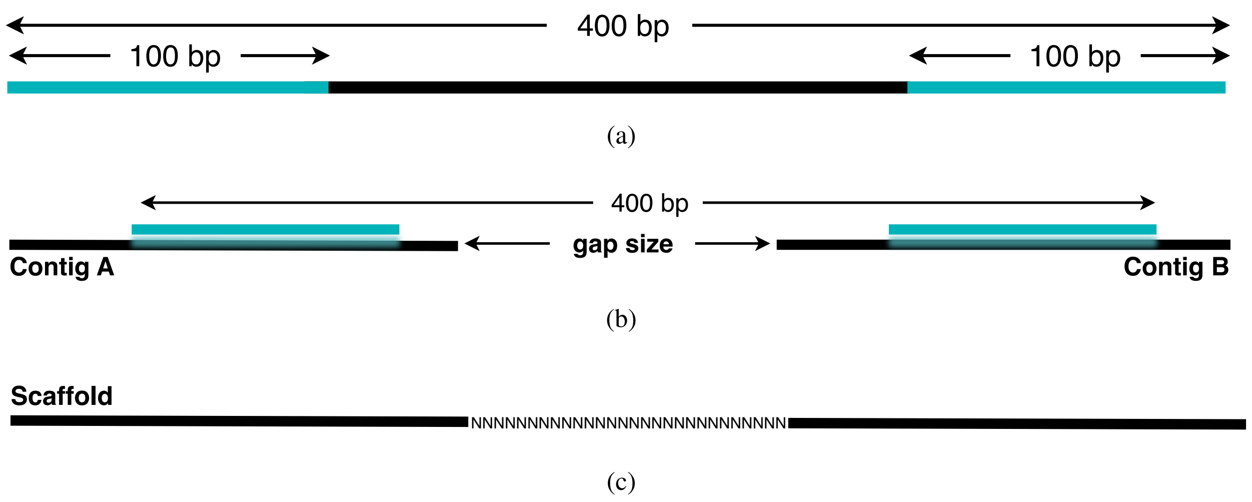


Figure 1.2: (a) Paired-end reads or mate-pairs are formed by sequencing the ends of a sequence of known length. (b) Because the orientation and distance of the paired-end reads is known, they can be used to position and orient contigs relative one to another. (c) The result is a reconstructed sequence composed of known and unknown regions. Unknown bases are denoted using the letter ‘N’.

shorter than normal unpaired DNA reads. However, the paired-reads are sequenced from either end of a longer *insert* sequence of known length using one of a number of paired-end sequencing technologies (see Figure 1.2a). Unique alignments of the paired reads to the pre-assembled contigs are supporting evidence for the inference of distance and orientation of contigs (see Figure 1.2b). Scaffolding thus aims to reconstruct the chromosomal sequences by orienting the contigs and fixing them at distances suggested by paired-end linkages (see Figure 1.2c). The *gaps* are reported using the inferred number of bases in the gap (denoted using the letter ‘N’). The scaffolding problem using paired-end data can likewise be modeled as a graph where the nodes are contigs and the edges are paired-end linkages between contigs weighted by the amount of evidence supporting the linkage.

The goal of scaffolding is to continue to properly orient and fix contigs at the correct distances until the number of scaffolds approaches the number of expected chromosomes. A full reduction to the exact number of chromosomes is rarely possible without substantial manual curation. Post-processing includes filling gaps between contigs using a variety of both wetlab and computational techniques.

1.2.3 The Challenge of Diploid Genome Assembly

Numerous assembly algorithms exist, each with its own unique strengths. Some algorithms work better with longer reads; others work well with short reads. Some algorithms are designed to work well with bacterial genomes, which by nature are quite different from multi-cellular, eukaryotic species. Algorithms can vary in the type of input they require and the type of output they produce. For example, some algorithms depend heavily on confidence scores for the base calls at each position in a sequenced DNA read, while others begin by assuming no errors in the DNA bases called by the sequencing machinery. For the most part it is assumed that the goal is to reproduce a single haplotype.

Most algorithms reasonably assume the reconstruction of a single haplotype because it is far simpler and historically biologists have catered to it. To understand how this assumption simplifies the problem, let us compare the reassembly of chromosomes from short DNA reads to reconstructing a jigsaw puzzle. In the case of a diploid species, reconstructing two different haplotypes would be like mixing the pieces of multiple similar jigsaw puzzles together before attempting a reconstruction of each individual puzzle. There is no clear way to identify to which puzzle each piece belongs and perhaps more frustrating, the pieces from both puzzles look (or sometimes are) identical. Rather than develop algorithms to handle this complexity, most sequencing projects elect to inbreed a species over multiple generations so that the haplotypes inherited at each generation are increasingly similar. This would be analogous to reducing our problem from reconstructing multiple mixed puzzles back to reconstructing a single jigsaw puzzle.

Though this assumption makes the problem easier for the bioinformaticians, the process of inbreeding in order to satisfy the assumption is not always an affordable luxury. Multiple iterations of inbreeding required to yield a highly homozygous genome can necessitate years of waiting, not to mention the increased cost of maintaining the project in the interim. In some cases, there are ethical concerns against inbreeding. Even with careful inbreeding, it is impossible to guarantee that two haplotypes are perfectly homozygous, and therefore some ambiguity is inevitable. In cases where the heterozygosity proves too great, additional expenditures may be required to obtain data for a more homozygous specimen. An additional potential drawback is that the species resulting from inbreeding is either not viable or does not represent the same organism that was initially being investigated. Hence, there is a critical need for sequencing algorithms capable of handling the complexity of heterozygous diploid genomes, a complexity that violates the fundamental assumption of the vast majority of traditional sequencing algorithms. Having matured past the infant stages of bioinformatic algorithms, it becomes necessary to abandon the simplifying and costly assumptions made in earlier stages in order to more efficiently and effectively solve whole-genome sequencing.

Of the two reconstruction phases, scaffolding is particularly impacted by the assumption of a single haplotype. The algorithm for extending reads into contigs remains largely unchanged even when a single haplotype is not assumed. Scaffolding algorithms, however, generally rely heavily on the assumption that most contigs belong in only a single scaffold, representative of a portion of a haploid genome. We propose the development of a scaffolding algorithm which accommodates the unique challenges of diploid genomes.

1.2.4 Thesis Statement and Document Organization

Ongoing research in the Computational Sciences Laboratory has for some time attempted to complete a sequencing and assembly of the *Rubus idaeus cultivar heritage* raspberry genome. For reasons of time and money, the samples collected for this species were not taken from an inbred organism and consequently we have data for a very heterozygous diploid organism. This master's thesis is motivated by the lack of sufficient tools for handling a genome of this type. Both the problem and the solution are readily generalizable to any sequencing project. *We hypothesize that adaptation of existing graph algorithms to the problem of diploid genome assembly will aid in the resolution of haplotype-specific variation.* Success in this endeavor promises to accelerate the rate at which new species can be sequenced, reduce costs, and eliminate the ethical dilemma of inbreeding. Improving the efficiency of sequencing will ultimately provide data collections of sufficient size to more easily study the effects of genetic variation.

The thesis is organized as follows. In chapter 2, we present related work in the field of heterozygous genome assembly. In chapter 3, we present a paper that was presented at BIOCOMP '12 which describes a scaffolding model that is built to allow for multiple haplotypes. This research provides the groundwork for chapter 4, in which we present a paper that was submitted to ACM BCB 2013. This chapter discusses a specific algorithmic challenge in scaffolding called the *contig orientation problem*. We show that the problem is readily reducible to a weighted MAX-CUT problem and demonstrate how solutions to this problem in diploid genome assemblies can be used to extract features that are unique to diploid heterozygous genomes. We present our own greedy heuristic solution and demonstrate its superior performance in reducing ambiguity in scaffold graphs. Solutions were assessed using four metrics: the total count of edges retained; the total weight of edges retained; the total count of edges excluded; and the total weight of edges excluded. We also use BLAST [1] to verify the accuracy of inverted-haplotype predictions, as described in more detail in that chapter.

Chapter 2

Related Work

Most widely-used scaffolding algorithms are explicitly designed to reconstruct homozygous genomes. The Newbler assembler, developed by 454 Life Sciences and distributed with 454 sequencing machines, has been used on a number of assembly projects [31, 37, 41]. Its efficiency in contigging is particularly notable given that it works natively with the .SFF data format to account for the specifics of pyrosequencing errors. Newbler requires uniquely mapping mate-pairs (i.e. paired-reads) as scaffolding evidence, disregarding reads which potentially map to multiple contigs. This can prove problematic with diploid genome reconstruction where reads may map to multiple homologous sequences.

Bambus [38] uses mate-pair information together with other types of linking data to infer the orientation and ordering of contigs to hierarchically construct scaffolds. The linkage data is used to create a graph where nodes are contigs and edges represent linkage evidence. Unlike many scaffolding algorithms, Bambus does not disregard ambiguous linkage evidence (e.g., multiply mapped pairs), and is capable of outputting pertinent data for manual finishing of ambiguous paths. However, this data is not used to inform the finishing algorithm in the case that automated finishing is required. Rather this simple greedy algorithm iteratively finds the longest non-self-overlapping path without consideration of graph structures characteristic of repetitive or polymorphic sequences. No performance data on heterozygous genomes is reported.

Arachne [4, 21] is a Whole Genome Shotgun (WGS) assembler which was adapted for use in the assembly of the heterozygous *Ciona savignyi* genome (details below) [48]. In the contigging phase, Arachne uses depth of coverage and the presence of conflicting links as evidence of repetitive regions in order to avoid erroneous extension of contigs. Contigs are also incorporated in the filling of intra-scaffold gaps. The algorithm does not natively allow for assembling multiple haplotypes.

A few algorithms have been developed specifically to handle highly heterozygous genomes, but the majority are designed in-house for specific problems without successive marketing or publication of the algorithmic details. Contigs for the highly heterozygous (7.1X sequencing depth, 0.4% heterozygosity) diploid fungal pathogen *Candida albicans* genome [22] were constructed from Sanger reads (10.9X) using PHRAP (www.phrap.org). The authors expected that application of PHRAP to a heterozygous genome would lead to misassembles because the algorithm assumes a single-copy sequence. The authors assumed that where apparent gaps came as a result of heterozygous sequence, and not from a lack of data, standard gap-closing experiments were unfit to resolve the genome. They describe a process of identifying heterozygous regions, aligning separated homologs using BLASTN alignments, and then reconstructing the two homologous supercontigs. Though the approach taken in assembling the *Candida albicans* genome addresses some of the traditional oversights in assembly of heterozygous genomes, it is designed for use with small genomes (*Candida albicans* is an estimated 14.8 kb) and no effort has been made to reproduce the algorithm, let alone with genomes of several hundred million base-pairs.

Vinson et al. present a method for assembling highly polymorphic diploid genomes and applied this method in their assembly of the heterozygous (4.6% average substitution rate, 60 times that in humans) sea squirt *Ciona savignyi* genome [48]. The estimated genome size was 190 Mb and the assembly used WGS libraries with 5-kb and 40-kb insert sizes. The method, which at its heart employs a "splitting rule" to keep haplotypes from joining in the contigging phase, attempts to assemble two haplotypes separately and then merge the assemblies by selecting a representative base at each locus. The merge criteria is concerned primarily with minimizing the number of inter-contig gaps and secondarily with minimizing the switches between haplotypes in scaffold reconstruction. The method was employed as an extension to the Arachne assembly program [4, 21]. They illustrate how their algorithm significantly improved the N50 contig and scaffold size and read usage with respect to the default algorithm.

In drafting a genome sequence for the highly heterozygous (1 SNP per .1 Kb and 1 in/del per 0.45 Kb) grapevine *Vitis vinifera*, Velasco, R. et al. constructed metacontigs (i.e. scaffolds) using paired reads matching to non-repetitive parts of the contigs [46]. They found that 11.2% of DNA on homologous chromosomes differ. The estimated genome size was 504.6 Mb. The project used a combination of Sanger (6.5X) and 454 (4.2X) reads. Paired reads were Sanger reads. Coverage (relative to average coverage) was used to identify and handle contigs representing repetitive sequence. They used Assemble (Myriad Genetics Inc., Salt Lake City, Utah) to assemble contigs. Neighboring contigs were linked using paired-reads only if each read uniquely mapped to a single contig sequence. The same method was used to sequence the domesticated apple (*Malus x domestica*) genome and the grape species [47].

Zharkikh et al. [50] discuss problems and solutions with sequencing highly heterozygous genomes, using *Vitis vinifera* as a model. Their work suggests that heterozygous genome assembly requires focus on obtaining sufficiently high coverage of the sequence with high-quality reads to accurately determine consensus and variant base calls. Both Zharkikh et al. [50] and Vinson et al. [48] indicate that although assembly of homozygous genomes should allow for some mismatches in overlapping reads, in reconstructing individual haplotypes only overlaps with near perfect identity need be considered. They discuss phasing techniques and note that dramatic improvements in haplotype resolution are achieved by employing a larger variety of clone size. Matching coverage was used to identify repetitive sequence and distinguish them from unique sequence.

Comparative analysis of scaffolding algorithms on heterozygous data remains largely unexplored. Donmez et al. [12] present an assembler for highly polymorphic genomes called Hapsembler. The method employs a kmer hashing technique to detect read overlaps and then uses a Naive Bayes probabilistic error correction procedure. A simplified mate pair graph is created via transitive edge reduction. Paths between mate pairs are detected, allowing for reconstruction of multiple haplotypes. They assess Hapsembler's assembly of simulated reads from the *Ciona savignyi* genome and demonstrate its ability to recover haplotype-specific blocks containing 300 or more adjacent SNPs in half of the assembled genome. Donmez et al. [13] more recently released Scarpa, a standalone scaffolding module, which they test on both a simulated diploid genome as well as datasets from *G. clavigera* and *E. coli* data.

The vast majority of published algorithms have been created to assume reconstruction of haploid genomes. The challenges posed by heterozygous diploid genomes violate this fundamental assumption and analyses that are undertaken using most state-of-the-art software result in fragmented and error-prone assemblies. As demonstrated in several of these related works, researchers are required to develop ad hoc work-arounds to remedy these effects. There is a pressing need for tools that are specifically designed to address the complexities of heterozygous diploid genomes. Adapting existing graph algorithms to the specific challenge of scaffolding diploid genomes has the potential for yielding improved assembly and analysis of such genomes. This thesis explores different options for creating scaffolds from heterozygous data and the theoretical tradeoffs of each approach.

Chapter 3

ScaffoldScaffolder: An Aggressive Scaffold Finishing Algorithm

This paper was presented at BIOCOMP'12 and published in conference proceedings, pp. 385-390

Abstract

With next generation sequencing technologies producing vast amounts of nucleotide data, it becomes imperative to streamline and automate the genome assembly process as much as possible. Contig scaffolding algorithms, ideally designed to reconstruct full chromosomes, more often tend to produce a still intractable number of disjoint sequences, requiring further manual finishing of the genome. To this end we present ScaffoldScaffolder, an aggressive automated scaffold finisher which further reduces the scaffold set using paired-end data. We evaluate the performance of ScaffoldScaffolder on Newbler scaffolds created from the *Rubus idaeus cultivar heritage* raspberry species. Further automated genome finishing methods are discussed.

3.1 Introduction

3.1.1 Motivation

Genetic variation is the root cause for numerous diseases or predispositions to life-threatening diseases such as cancer and heart disease. Genetic variation in plants is the basis for variability in crop yields, nutritional value, and flavor. The future of scientific study in these areas depends heavily on the ability to study and characterize genetic variation.

Despite the direct bearing that genetics has on each of these instances, the specific genetic variations at play are not well-characterized, their effects are not well-understood, and the ability to scientifically study them is limited. To a large extent this is due to the relatively sparse amount of data that is available. This shortage derives in large part from the cost-prohibitive and somewhat unrefined nature of the technology and software used to obtain and analyze genetic data. The first human genome was sequenced less than 10 years ago and cost upwards of 3 billion dollars. Though DNA sequencing costs have decreased significantly, the time and manual effort required to produce finished genome sequences are still very restrictive. Despite global efforts to collect and sequence any and all forms of life, only about 1,200 organisms have been sequenced, most at a primitive level, hardly enough to begin to adequately characterize the patterns responsible for genetic variations of interest[1]. In order to deduce and characterize the effects of genetic variation, we need a larger number of high-quality sequenced genomes, implying the need for improved technology and software to produce them.

To this end we have undertaken to develop ScaffoldScaffolder, an automated scaffold finisher.

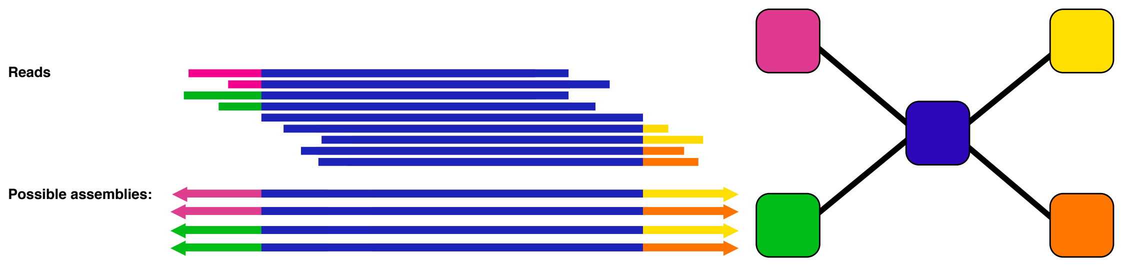
3.1.2 Background

Technology has thus far been only moderately successful at solving the problem of genomic sequencing. The most prominent methods require large-scale replication of genetic material which is then broken through sonication into an amalgam of short fragments of various sizes (called *reads*). From this mixture are extracted sequences suitable to the sequencing capacity of sequencing machines. The most cost-effective machines are capable of sequencing reads of approximately 100 bases while maintaining reasonably low error rates. Reads as long as 600 base pairs can be sequenced at a much higher price and with slightly higher error rates. In any case, the sequencers are unable to sequence anything that even begins to approximate the size of an entire chromosome which, for example in a raspberry, averages lengths of several million base pairs. The algorithmic challenge is to reassemble the full-length chromosomes from short DNA reads. The genome

[1]http://www.ncbi.nlm.nih.gov/About/tools/restable_mol.html

```
Read 1:       ACCCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGC
Read 2:        CCCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCA
Read 3:         CCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAG
Read 4:          CGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGA
Read 5:           GGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAA
Read 6:            GCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAAG
Read 7:             CGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAAGC
...

Consensus:    ACCCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAAGC...
```

(a)



(b) (c)

Figure 3.1: (a) Short reads whose sequences overlap are overlaid such that their consensus is a reconstruction of the original sequence from which the reads are taken. (b) Repetitive regions whose length exceeds that of sequenced reads create different possible reconstructed consensus paths. (c) The different reconstructions can be modeled as a graph where unambiguous consensus sequences are collapsed into nodes and evidence for the different paths are represented as edges.

reconstruction is divided into two phases: the overlapping of reads to form consensus contigs and the scaffolding of contigs to form chromosomes.

In the initial phase of the *assembly* process reads are used to form long contiguous sequences of known bases. This is accomplished by combining overlapping reads to produce longer consensus sequences called *contigs* (see Figure 3.1a). If all read-length genomic sequences were unique, we could continue this process until we reconstructed the original chromosomal sequence in its entirety. However, due to the presence of repetitive regions throughout a genome, reads will exist which support multiple paths of reconstruction (see Figure 3.1b).The ambiguity of this result is often modeled as a graph where the nodes are the unambiguous consensus contig sequences produced from combining overlapped reads and the edges are possible ways in which these contigs could be sequentially combined (see Figure 3.1c). Often the number of contigs can outnumber the actual number of chromosomes by as much as a factor of 10^3. The graph will often be missing nodes or edges due to insufficient coverage of certain areas of the genome or by erroneous contigs produced from errors during the read-sequencing phase.

Scaffolding is the step in the assembly process where additional information is leveraged to infer the relative distance and orientation of contigs. This is most commonly done using *paired-end data*. Paired-end data consists of pairs of short reads whose distance and orientation is known from the technique used to sequence them. Due to the read-size constraints mentioned above, the paired-reads are the same length or shorter than normal unpaired DNA reads. However, the paired-reads are sequenced from either end of a longer *insert* sequence of known length using one of a number of paired-end sequencing technologies (see Figure 3.2a). Unique mappings of the paired reads to the pre-determined contigs are supporting evidence for the inference of distance and orientation of contigs (see Figure 3.2b). Scaffolding thus aims to reconstruct the chromosomal sequences by orienting the contigs and fixing them at distances suggested by paired-end linkages (see Figure 3.2c). The *gaps* are reported using the inferred number of bases in the gap (denoted using the letter 'N'). The goal of scaffolding is to continue to properly orient and fix contigs at the correct distances until the number of scaffolds approaches the number of expected chromosomes. The quality of an assembly notably increases by using a large variety of clone sizes in the scaffolding phase [50]. However, additional measures are required to reduce the resulting scaffold number to the chromosome number.

We have developed ScaffoldScaffolder, a lightweight tool designed to automate the scaffolding of scaffolds using paired-end data. Whereas it is the purpose of a scaffolder to recover the orientation and placement of contigs inasmuch as the data will accurately allow, the purpose of the ScaffoldScaffolder is to act as a post-processing step to aggressively reduce the number of sequences as much as possible by leveraging remaining unused linkages inferred from paired-end data. Rather than simply concatenating resulting scaffolds in random order, at random distances, and in random orientations, ScaffoldScaffolder attempts to infer the correct scaffolding, though in a somewhat less cautious manner than a scaffolder.

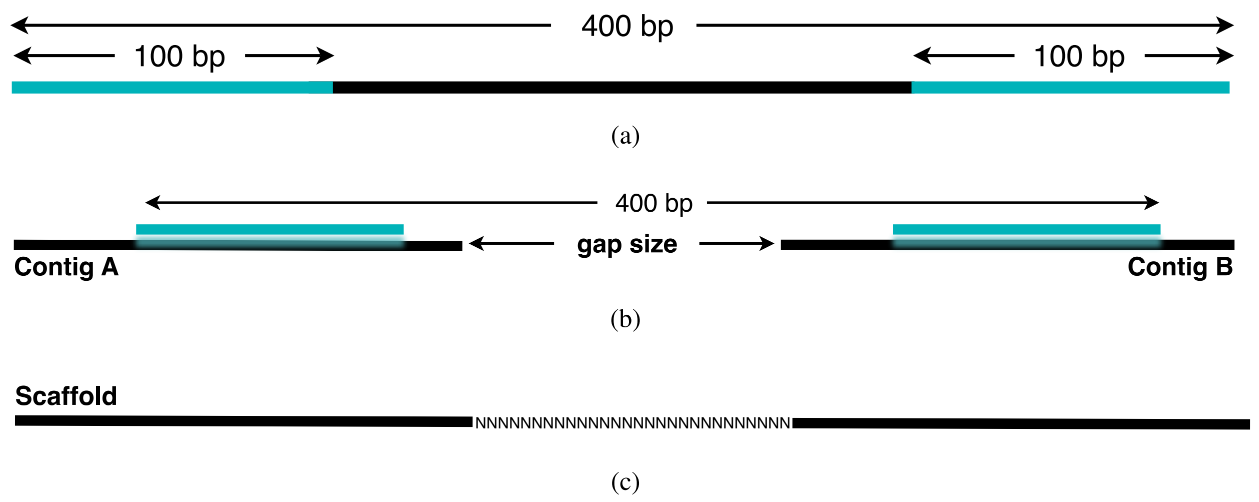


Figure 3.2: (a) Paired-end reads or mate-pairs are formed by sequencing the ends of a sequence of known length. (b) Because the orientation and distance of the paired-end reads is known, they can be used to position and orient contigs relative one to another. (c) The result is a reconstructed sequence composed of known and unknown regions. Unknown bases are denoted using the letter 'N'.

3.2 Related Work

The Newbler assembler, developed by 454 Life Sciences and distributed with 454 sequencing machines, has been used on a number of assembly projects [31, 37, 41]. Its efficiency in contigging is particularly notable given that it works natively with the .SFF data format to account for the specifics of pyrosequencing errors. Newbler requires uniquely mapping mate-pairs (i.e. paired-reads) as scaffolding evidence, disregarding reads which potentially map to multiple contigs.

Bambus [38] uses mate-pair information together with other types of linking data to infer the orientation and ordering of contigs to hierarchically construct scaffolds. The linkage data is used to create a graph where nodes are contigs and edges represent linkage evidence. Unlike many scaffolding algorithms, Bambus does not disregard ambiguous linkage evidence (for example multiply mapped pairs), and is capable of outputting pertinent data for manual finishing of ambiguous paths. However, this data is not used to inform the finishing algorithm in the case that automated finishing is required. Rather this simple greedy algorithm repeatedly finds the longest non-self-overlapping path without consideration of graph structures characteristic of repetitive or polymorphic sequences.

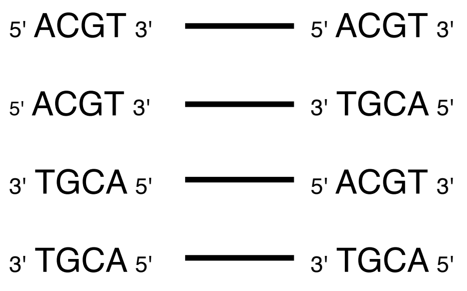
Arachne [4, 21] is a Whole Genome Shotgun (WGS) assembler which has been used to assemble heterozygous genomes [48]. In the contigging phase, Arachne uses depth of coverage and the presence of conflicting links as evidence of repetitive regions in order to avoid erroneous extension of contigs. These contigs are incorporated in filling intra-scaffold gaps.

SOAPdenovo is a short-read assembly algorithm developed by the Beijing Genomics Institute (BGI) which has been employed in a large number of genome projects [29]. The program is designed primarily to function with Illumina GA short reads in reconstruction of large genomes.

MAIA [35] integrates multiple de novo and comparative assemblies by creating a graph of the contigs from these assemblies and their alignments. Four properties for the edge weighting are implemented, namely contig length, overlap length, length of non-aligned overhang, and original assembly quality. This approach makes it possible to use specific assemblers for different next-generation data sources and enables the use of multiple known related genomes in the assembly process. The algorithm was applied on the de novo sequencing of the *Saccharomyces cerevisiae* and demonstrated improvements upon single assembly methods (Velvet, Celera, MAQ) and other hybrid methods (Velvet, Minimus). The disadvantages are that MAIA inherently relies on a very closely related genome in the assembly process and the computational expense of the algorithm renders the approach impractical for larger genomes. The algorithm, like many, is designed for use with homozygous genomes.

3.3 Methods

ScaffoldScaffolder is designed to be used as an iterative algorithm where each successive iteration utilizes a paired-end library of a larger insert size than the previous iteration. Each iteration requires as input a series of sequences to be scaffolded in fasta format and any number of similarly-sized paired datasets. The high-level purpose of the algorithm is to use the paired datasets to infer scaffoldings of the input sequences and then to select and output an unambiguous subset of the scaffoldings in fasta format. It additionally outputs information detailing the specifics of the input sequences which compose the new scaffolds.



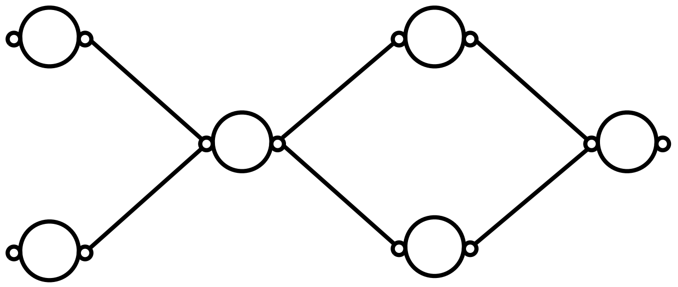


Figure 3.3: (a) There are four possible ways for two sequences to be adjacent. The correct orientation can be uniquely defined by specifying which ends are adjacent. (b) Orientation can be preserved in a graph model using bi-terminal nodes where each terminal represents a sequence end.

Internally the algorithm stores input sequences in the context of a graph where nodes represent sequences and edges between nodes indicate that paired data exists to suggest that two sequences should be scaffolded. We must make a slight modification on how we define nodes in the context of this problem. In classic graph theory, we say that if (u,v) is an edge in a graph G = (V,E), then node v is adjacent to node u. However, in the context of our problem it is possible for two sequences to be adjacent in one of four different orientations. One possible solution to this problem relies on the biological concept of sequence orientation which defines one end of the sequence as the 5' (said *five prime*) or *upstream* end and the opposite end of the sequence as the 3' or *downstream* end inasmuch as DNA synthesis proceeds in a 5' to 3' direction. This directionality is an inherent characteristic of each sequence.

One way to uniquely distinguish between the four different orientations of two DNA sequences is to specify which two ends are adjacent (see Figure 3.3a). We can model this in our graph by defining our nodes as *bi-terminal*, where edges to one terminal represent adjacency to the 5' end of the represented sequence and edges to the second terminal represent adjacency to the 3' end of the same sequence (see Figure 3.3b). This concept of a graph can be reduced to the standard definition of a graph by making each terminal its own node and creating an edge between them.

In the ScaffoldScaffolder, this *scaffold graph* is initialized only with the sequence nodes; edges are later progressively added as each input dataset is processed for linking evidence. It is assumed that contigs represent unique sequences and thus there is a one-to-one relationship between nodes and sequences.

The algorithm uses an external mapping algorithm, Bowtie [28], to map reads in the input paired datasets to the sequences to be scaffolded. While the algorithm is heavily modularized to support other mapping algorithms (including GNUMAP [9] and BLAST [1]), testing has been limited to Bowtie. Experimentation to date has required mappings to be unique (meaning no more than a single alignment location exists for the mapped read) in order to maximize confidence in the resulting scaffolds. ScaffoldScaffolder currently gives the user the option of adjusting this parameter as well as parameters dictating read-trimming options, alignment-mismatch options, and options for skipping the first *n* reads in a dataset. A parameter allows the user to specify the paired-end orientation either as -fr (Illumina paired-end protocol), -rf (Illumina mate-pair protocol), or -ff (454 mate-pair protocol).

From the Bowtie results ScaffoldScaffolder then identifies pairs for which both ends are uniquely mapped. In cases where both ends map within the same sequence, the distance between the mappings is cataloged in order to infer an insert size for the library. In cases where ends map to distinct sequences, the algorithm infers the orientation and gap size between the two base sequences. Assuming that the gap size is viable (i.e. nonnegative), the weight of the corresponding edge in the scaffold graph is linearly incremented and the inferred gap size for the scaffolding of the two oriented base sequences is cataloged. The final gap size is the mean of the inferred gap sizes.

The process of mapping paired-reads and then loading the scaffold graph according to the mapping results is repeated for each provided paired-end source in the respective iteration of the algorithm. At the conclusion of this phase, the scaffold graph contains a number of ambiguous linkages where a given base sequence may have multiple possible scaffoldings in the upstream and/or the downstream direction. ScaffoldScaffolder assumes that a given base sequence will be scaffolded with only one sequence in either direction. In order to reduce the graph to include an unambiguous subset of scaffold edges, the edges are sorted by weight, following which edges are greedily considered for inclusion in the final graph. If adding an edge creates an ambiguous scaffolding, the edge is skipped. A minimum support parameter determines the minimum number of unique pairs required as support for an edge to be included.

Scaffold sequences are constructed from the disambiguated scaffold graph and these sequences, together with any unscaffolded sequences, are output in fasta format by decreasing order of length.

3.4 Results

We tested the ScaffoldScaffolder algorithm on Newbler scaffolds created for the heterozygous *Rubus idaeus cultivar heritage* raspberry genome.

Contigs were first assembled from the reads using the Newbler assembler. Due to memory and time constraints, the 5k dataset was not incorporated into the Newbler assembly. Aside from this exception, the same data used in the Newbler assembly was used as input to ScaffoldScaffolder.

The assembler parameters were set to require a minimum length of 30 bases, a minimum overlap length of 70 bases, and a minimum overlap identity of 98 bases. The large genome assembly, heterozygotic, and scaffold flags were enabled. Using these parameters, Newbler produced 123,121 contigs and 13,037 scaffolds.

ScaffoldScaffolder was parameterized to use Bowtie for the mapping of paired reads, with a maximum of 3 mismatches, and only allowing uniquely mapping reads. The minimum support required for valid links was 1.

ScaffoldScaffolder was able to reduce the scaffold count by 5399, representing a reduction of over 40% of the scaffolds produced using Newbler's scaffolding algorithm alone (see Table 3.1).

The 3kb and 20kb datasets (those produced using the 454 mate pair protocol) had noticeably lower rates of alignment. We suspect this derives from the inability of the Bowtie aligner to consider insertions or deletions when aligning reads. This proves troublesome for reads sequenced using the 454 protocol which often have insertions and deletions in homopolymorphic sequences. Consequently, selection of other short-read mapping algorithms capable of handling indels could further improve the performance of the ScaffoldScaffolder algorithm.

Table 3.1: Reduction of Newbler Scaffolds via Multiple Iterations of ScaffoldScaffolder

Iteration (insert size)	Reads Uniquely Aligned	Scaffold Count	Max Scaffold Size	Avg Scaffold Size
Initial		13,037	4,456,429	19,313
400b	171,490,959	11,620	4,456,429	21,905
3kb	397,429	11,271	4,456,429	22,643
5kb	99,333,568	8,695	4,456,429	30,976
20kb	893,745	7,638	4,678,214	37,961

3.5 Discussion

ScaffoldScaffolder attempts to provide an algorithmic solution to automated finishing using paired-end data. Although it may be argued that the aggressive nature of the algorithm will lead to inaccuracies in the resultant assembly, similar inaccuracies are common to other prevalent finishing methods of which we will briefly discuss two.

3.5.1 Genetic Linkage Map

Biological assays are capable of inferring the relative distance along chromosomes of a number of specific genetic sequences based on what is called the *recombination rate* of protein-coding sequences (i.e. genes). Recombination refers to the rearrangement and exchange of genetic material that occurs when chromosomes cross over one another. The likelihood of such a rearrangement occurring between two genes, known as the recombination rate, increases as a function of the distance between the two genes. Thus the relative distance and ordering of certain observable genes can be inferred biologically in order to create a *genetic linkage map*. These genes, whose sequences are known, can be used to guide the finishing of the assembly. Assuming that such a genetic linkage map is available, this process is quite accurate, but may still fail to place a number of scaffolds.

3.5.2 Related Genomes

A second approach to genome finishing attempts to infer the distance and orientation of contigs by using the known sequence of a closely related genome. We refer to the degree of genetic similarity in gene-order between different species as *synteny*. To the extent that the genomes of two species are syntenic, the ordering and orientation of similar sequences on the related genome can be used to guide the assembly of the target scaffolds. The challenge with this approach is proper identification and treatment of genomic differences.

3.6 Conclusion

In this research, we present ScaffoldScaffolder, an aggressive automated scaffold finisher. We have illustrated its effectiveness in significantly reducing a set of Newbler scaffolds created for the *Rubus idaeus cultivar heritage* raspberry genome. Future development aims to address the complexities of scaffolding heterozygous genomes with inclusion of structural/sequence-based heuristics to identify and assemble distinct haplotypes. Improved input data analysis will aim to infer parameters so as to reduce the information required from the user for execution.

Chapter 4

Solving the Contig Orientation Problem via Bidirected Graph Reduction to a Directed Graph

A shortened version of this paper was submitted to ACM Conference on Bioinformatics, Computational Biology and Biomedical Informatics (ACM BCB 2013). The analysis section was not included in the shortened version.

Abstract

In the context of genome assembly, the contig orientation problem is described as the problem of removing sufficient edges from the scaffold graph so that the remaining subgraph assigns a consistent orientation to all sequence nodes in the graph. This problem is equivalent to finding the maximum fully-directed subgraph of a bidirected scaffold graph. We show how this problem is equivalent to both the weighted MAX-2-XORSAT and weighted MAX-CUT problems and provide reduction algorithms. A linear-time algorithm is presented which solves this problem by heuristically maximizing total edge weight. We compare the performance of this algorithm to a random solution and to a semi-definite programming solution. We show how inverted repeats and inverted haplotypes violate the fundamental assumption of the contig orientation problem and demonstrate how solutions to the contig orientation problem can be used to identify these biologically significant regions.

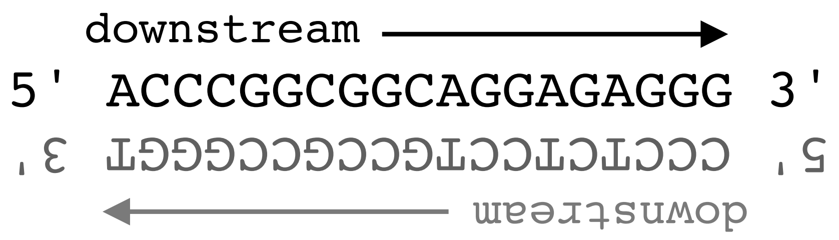


Figure 4.1: By convention DNA is written 5’ to 3’. DNA is double-stranded with 5’ to 3’ directionality of the reverse-complement strand being opposite that of the forward strand.

4.1 Introduction

Accurate and efficient genome assembly algorithms are essential in unlocking the solutions to challenges posed by genetic disease, genetic engineering, and even next-generation digital information storage [8]. The term *genome* describes a complete set of DNA in the cell of an organism. Within the genome, DNA is organized as a distinct set of molecules called *chromosomes*, the number of which is a unique characteristic of a species. Each chromosome is composed of a sequence of *nucleotide bases*, or simply *nucleotides*, which encode all of the functionality of a living organism. The goal of genome assembly is to ascertain the identity of this sequence and is prerequisite to making inferences about the complex mechanisms that govern life.

DNA is inherently directional, which means that defining the head of a sequence (called the 5’ or *five prime* end) and the tail of a sequence (3’) is as essential as defining the nucleotide residues themselves. By convention, the *forward strand* of a sequence s^+ is written in the 5’ to 3’ direction (the same order in which DNA is biologically replicated, transcribed, and sequenced); however, as DNA is *double-stranded*, it is always implied (though usually not written) that for any sequence s^+, an equally viable reverse-complement sequence s^- exists whose 5’ to 3’ direction is opposite that of s^+ (see Figure 4.1). The forward strand is commonly referred to without the superscript. We will define moving in the 5’ to 3’ direction as moving *downstream* and 3’ to 5’ as moving *upstream*.

Next-generation sequencing technologies are currently only capable of *sequencing* (i.e., ascertaining the sequence of nucleotide residues of) short DNA fragments called *reads*. Consequently

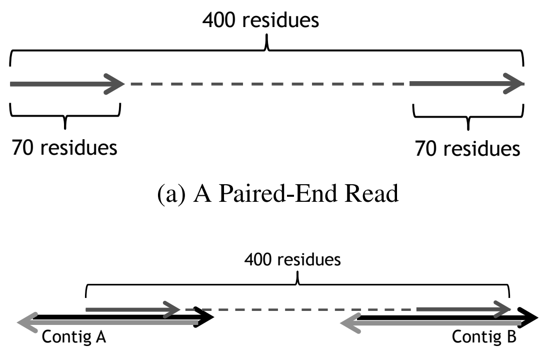
```
Read 1:     ACCCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGC
Read 2:      CCCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCA
Read 3:       CCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAG
Read 4:        CGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGA
Read 5:         GGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAA
Read 6:          GCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAAG
Read 7:           CGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAAGC
…

Consensus:  ACCCGGCGGCAGGAGAGGGGATGAAGATGGCGGACGCGAAGCAGAAGC…
```

Figure 4.2: The contigging phase overlaps short DNA reads to determine the consensus sequence of the larger derivative molecule.

the genome assembly process requires first fragmenting chromosomes (which in humans are on the order of hundreds of millions of nucleotides in length) into small segments to be sequenced. Then confident overlaps are found between the reads to recover the chromosomal sequence. This process is known as *contigging* (see Figure 4.2).

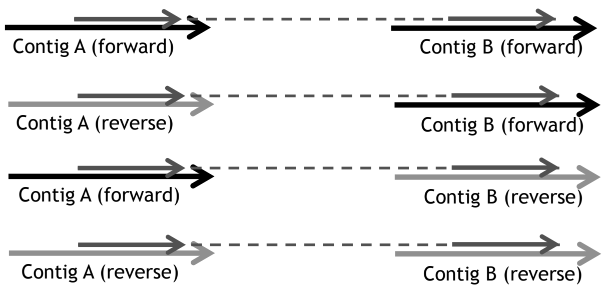
In reality, insufficient molecular sampling and repetitive regions in the DNA prevent full chromosomal reconstruction and it is thus commonplace for assembly algorithms to produce a large set of partially-reconstructed chromosomes termed *contigs*. Contigs must then be oriented and positioned relative to one another in order to reconstruct full chromosomes through *scaffolding*. Scaffolding uses longer fragments of a known length whose ends are sequenced (called *paired-end reads*) to infer positional and orientational relationships between contigs (see Figure 4.3). Considering that a paired-end may match to either the forward sequence or the reverse-complement sequence, there are four possible ways in which two adjacent contigs may be oriented relative to one another (see Figure 4.3c).

The problem of scaffolding contigs is commonly modeled as a graph (i.e., a *scaffold graph*) where vertices are contigs and edges indicate candidate scaffoldings of contigs, weighted by the amount of supportive evidence for the scaffolding. It should be noted that in the contigging phase, reads from repetitive regions in the genome will overlap and combine into a single contig. It is not uncommon for scaffolding algorithms to ignore these *repeat contigs*, as proper repeat resolution



(a) A Paired-End Read

(b) Scaffolding



(c) Orientation of Contigs in Scaffolds

Figure 4.3: a) A paired-end read is a long DNA fragment whose ends have been sequenced (arrow indicates the 5' to 3' directionality). b) Paired-end reads are used to infer the relative position and orientation of contigs (contigs shown with reverse-complement in gray) to which the ends find matching locations. c) Although the paired-end reads have a fixed position and orientation, depending on whether an end aligns to a contig's forward or reverse strand, contig pairs may be oriented in four different ways within a scaffold (gray contigs represent 5' to 3' of the reverse-complement strand).

often requires manual curation [38]. Repeat contigs naturally suggest varied scaffoldings that create conflicts in the scaffold graph. Sequencing errors create additional complexity, suggesting contig positionings and/or orientations that are inconsistent with other scaffolding evidence. As a result of these complications, the problem is generally simplified, so that the goal simply becomes finding a maximal path through the graph that incorporates each contig sequence once. By definition, including each sequence once requires first assigning each contig a single orientation in the reconstruction.

The *contig orientation problem* describes the challenge of assigning contig orientations so as to minimize conflicting orientation evidence [38] . More specifically, the goal is to remove the minimum number of edges from the scaffold graph so that the remaining subgraph suggests a single consistent orientation of all vertices in the graph (see Figure 4.4).

One purpose of the current paper is to formally define the contig orientation problem and compare possible solutions. However, a second and equally important purpose is to discuss the viability of the problem's foundational assumption: that each non-repetitive sequence belongs in the reconstruction in a single orientation. We will refer to this as the *single-orientaton assumption*.

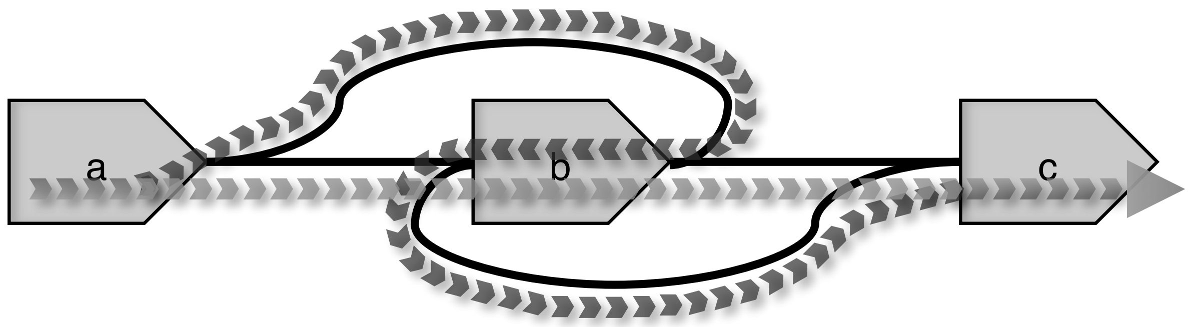


Figure 4.4: *The Contig Orientation Problem.* Shown are three contigs, A, B, and C and edges (solid lines) between them suggesting ways in which these contigs might be relatively positioned and oriented in molecular reconstruction. Note that two edges support scaffolding sequences A^+, B^+, C^+ (light gray arrows). Two other edges suggest scaffold A^+, B^-, C^+ (dark gray arrows). The two possible reconstructions suggest internally inconsistent orientations of contig B relative to contigs A and C. The contig orientation problem is the problem of excluding the minimum number of edges (or minimum edge weight) so that a single orientation is suggested for all contigs in the graph.

Ultimately we find that the assumption fails, even when repeats are screened; however, in considering the cases in which this assumption fails, we discover a possible method for detecting two biologically significant structures: inverted repeats and inverted haplotypes.

The paper is outlined as follows. We begin by formally defining the problem in terms of bidirected graph theory, demonstrating an equivalency to the well-studied weighted MAX-2-XORSAT and weighted MAX-CUT problems. We then present a novel solution to which we compare a number of weighted MAX-CUT solutions. Finally, we comment on the validity of the single-orientation assumption and its role in identifying inverted repeats and inversion variants.

4.2 Analysis

In this section we formally define the contig orientation problem in terms of bidirected graph theory. We demonstrate its equivalence with MAX-2-XORSAT and MAX-CUT.

4.2.1 Bidirected Scaffold Graphs

A *weighted bidirected graph* (as introduced by [14]) is formally defined as an undirected multigraph G with a set of vertices V and a set of *bidirected edges* E. A bidirected edge e has two *endpoint orientations*, one with respect to each of its endpoints. An endpoint orientation may be either *positive* or *negative*, defining e as either *positive-incident* or *negative-incident* to an endpoint. We further associate a weight function with the edges, $w{:}E{\mapsto}\mathbb{R}^+$. We say that a bidirected graph is *connected* if its underlying undirected graph is connected.

In the graphical representation of a bidirected edge e, we represent positive-incidence with an arrow pointing out of the vertex and negative-incidence with an arrow pointing in to the vertex. Based on this representation, we say that e is: *directed* if it is positive-incident to one endpoint and negative-incident to the other; *introverted* if positive-incident to both endpoints; and *extraverted* if negative-incident to both endpoints (see Figure 4.8). A *directed graph* is a special case of a bidirected graph in which all edges are directed edges.

A valid (v_1,v_k)-*walk* is a sequence $v_1,e_1,\ldots,v_{k-1},e_{k-1},v_k$ where e_i is an edge incident to v_i and v_{i+1} and for all $2 \leq i \leq k-1$, e_{i-1} and e_i have opposite endpoint orientations incident to v_i (see Figure 4.6).

Based upon this definition, we define a *bidirected scaffold graph* for a set of contigs C and a set of weighted candidate scaffoldings F as a bidirected graph $G{=}(V,E)$ in which vertex $v_i \in V$ represents contig $c_i \in C$ and a weighted bidirected edge e linking vertices v_i and v_j represents the weighted candidate scaffolding $f \in F$ between contigs c_i and c_j (see Figure 4.5). The endpoint orientations of e are determined by the relative orientation of the forward strands of c_i and c_j in f: if the forward strands of c_i and c_j are oriented in the same direction, then e is a directed edge that is positive-incident to the vertex representing the upstream contig; if the forward strands are oriented away from one another (i.e., 5' ends are proximal), then e is an extraverted edge; and if the forward strands are oriented towards one another (i.e., the 3' ends are proximal), then e is an introverted edge (see Figure 4.8).

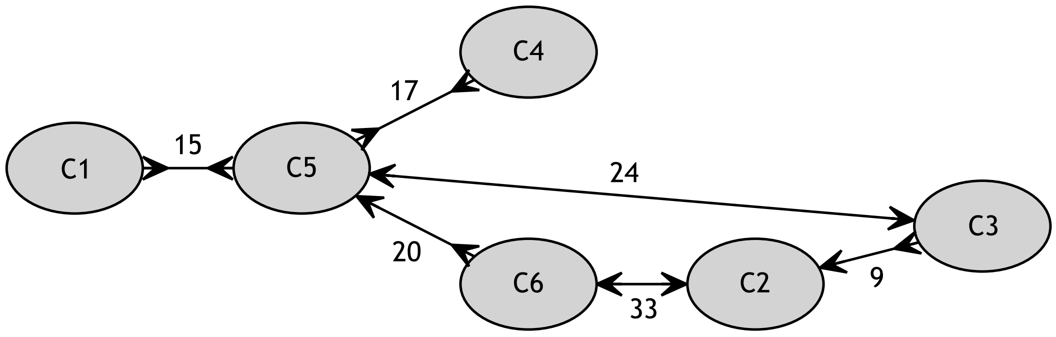


Figure 4.5: *Bidirected Graph Notation.* A scaffold graph constructed in bidirectional graph notation in which nodes are contigs and edges are candidate scaffoldings of contigs, weighted by the amount of scaffolding evidence. In this notation, relative sequence orientation is notated in the edge forms.

Two critical specifications must be made at this point to the bidirected scaffold graph in order that our biological constraints are maintained. First, the 5' to 3' directionality of a DNA molecule must be consistent along the entire length of the sequence. This means that introverted and extraverted edges, both of which represent internally inconsistent 5' to 3' directionality of the forward strand, violate a biological constraint. As per this definition, only directed edges are considered *valid* in the final scaffold reconstruction. The second specification, however, is that because the biological molecule allows us to consider a contig c_i as either its forward strand c_i^+ or its reverse-complement strand c_i^-, for any vertex $v_i \in V$ we can arbitrarily select between the *forward-orientation assignment* v_i^+ and the *reverse-orientation assignment* v_i^- in order to ensure consistency of the 5' to 3' directionality of contigs c_i and c_j in scaffolding f. We will refer to this selection as the *contig-orientation assignment* of c_i or *vertex-orientation assignment* for v_i. Furthermore we refer to a contig-orientation assignment for all contigs in C (or vertices in G) as a *contig-orientation assignment* of C (or *vertex-orientation assignment* of G). Switching between assignments v_i^+ and v_i^- is equivalent to flipping all endpoint orientations incident to v_i (see Figure 4.6c), thus potentially rendering introverted and extraverted edges as directed edges, or vice versa. We will refer to a vertex v_i with possible vertex-orientation assignments v_i^+ and v_i^- as an *orientable vertex*. As the

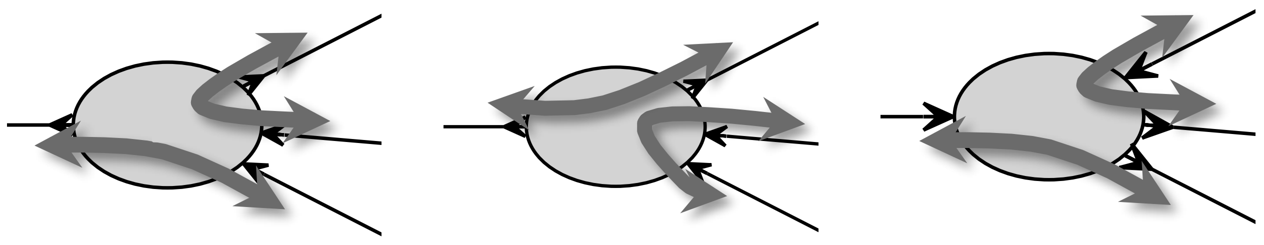

(a) Valid Walk (b) Invalid Walk (c) Rev. Orientations

Figure 4.6: (a) In a bidirected scaffold graph, a valid walk enters and exits a node through oppositely-oriented arrows. (b) An invalid walk enters and exits a node through same-oriented arrows. (c) Reversing all edge-orientations adjacent to a node results in the same potential valid walks. This is biologically equivalent to selecting the reverse-complement s^- in place of the forward sequence s^+.

forward strand c_i^+ of each assembled contig c_i is arbitrarily given as input, each corresponding vertex v_i is initially assumed to be assigned the vertex-orientation v_i^+.

A final clarification must be made prior to formally defining the contig orientation problem. The goal of finding a maximal contig-orientation assignment for a graph is not equivalent to the goal of finding a maximal walk through the scaffold graph (i.e., creating linear scaffolds). There are several cases (e.g., heterozygous sequences and/or repetitive sequences on different chromosomes) in which a single connected contig-graph component can account for *multiple* walks resulting in multiple scaffolds that belong in the final reconstruction [39]. Indeed further algorithmic elucidation of reconstructive paths is required following the resolution of the contig orientation problem. What *will* be guaranteed from resolution to the contig orientation problem is that any valid walk chosen from the resulting subgraph will have consistent internal orientation of all constituent contigs. A maximal solution to the contig orientation problem seeks to accomplish this while maximizing the scaffolding evidence that is rendered valid by the orientation assignment.

4.2.2 MAX-DIR-SUBGRAPH Problem

We formally state the corresponding decision problem as follows:

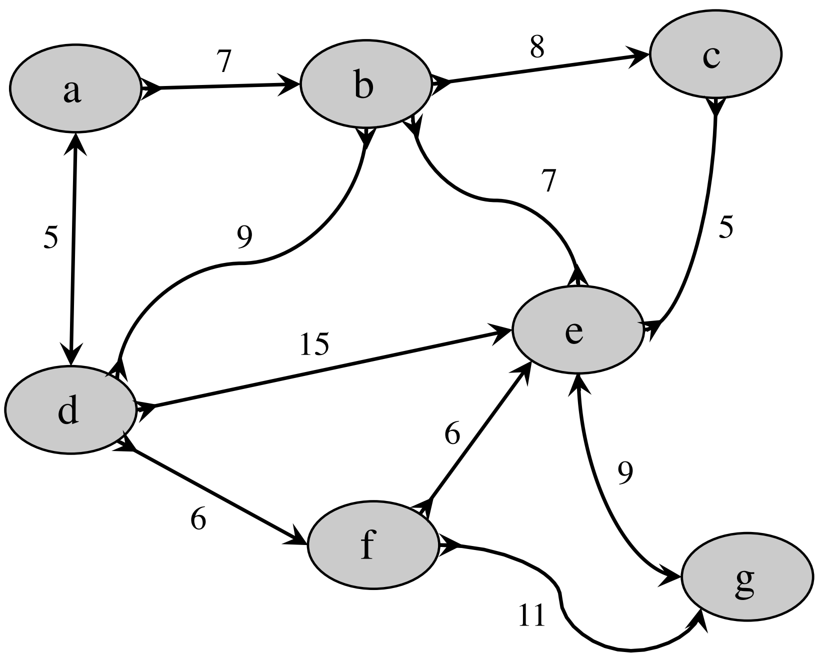


Figure 4.7: An example of a bidirected scaffold graph. The MAX-DIR-SUBGRAPH problem is defined as the problem of finding a vertex-orientation assignment which maximizes the number of directed edges.

MAX-DIR-SUBGRAPH $= \{(G,k) \mid G$ is a bidirected graph with orientable vertices, and there exists a vertex-orientation assignment for G resulting in a subgraph containing at least k directed edges$\}$

Depending on whether the preferred bias is towards more evidence or more edges, the weighted and unweighted versions of this problem (respectively) become important. In the following equivalence proofs of the MAX-DIR-SUBGRAPH problem to other well-known decision problems, we will consider the unweighted version and assume that the weighted version is readily deducible from the unweighted. For the purposes of illustration, we include an example of a bidirected scaffold graph for which we would like to find a vertex-orientation assignment (see Figure 4.7) and show the corresponding MAX-2-XORSAT and MAX-CUT problems. The example demonstrates the weighted versions of each problem.

We first show that MAX-DIR-SUBGRAPH is equivalent to the MAX-2-XORSAT problem and then show that this latter problem is equivalent to the widely-studied MAX-CUT problem, which is NP-complete. Because the MAX-CUT problem is NP-complete, showing these equivalences proves not only that the MAX-DIR-SUBGRAPH problem is NP-complete, but also that heuristic solutions to the MAX-CUT problem can be applied to the MAX-DIR-SUBGRAPH problem. The same is true of the weighted forms of these decision problems.

4.2.3 Equivalence to MAX-2-XORSAT

Showing the equivalence of the MAX-DIR-SUBGRAPH problem to the MAX-2-XORSAT problem is straightforward and is also helpful in showing equivalence to the MAX-CUT problem. Let a *binary variable* be a variable whose value is either 1 (i.e., *true*) or 0 (i.e., *false*). A *literal* (which we will denote using x_i) is a binary variable b_i or its negation $\neg b_i$. A *2-XOR-clause*, or simply a *clause*, C, is a linear equation of 2 literals of the form

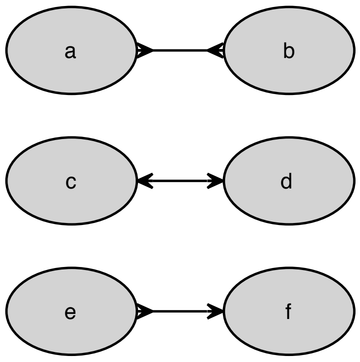
$$C = (x_i \oplus x_j)$$

A *truth assignment* is a mapping I that assigns 0 or 1 to each variable in its domain. The clause C is *satisfied* iff it evaluates to *true*, that is iff $(I(x_i) + I(x_j)) = 1$. A *2-XOR-formula*, or simply a *formula*, is a conjunction of distinct 2-XOR-clauses. A truth assignment satisfies a formula ϕ iff it satisfies every clause in ϕ. The decision problem corresponding to the MAX-2-XORSAT problem is thus stated:

MAX-2-XORSAT $= \{(\phi, k) \mid \phi$ is a 2-XOR-formula, and there is a truth assignment that satisfies at least k clauses$\}$

The MAX-DIR-SUBGRAPH problem can be reduced to the MAX-2-XORSAT problem as follows. Given a bidirected graph G with a set of orientable vertices V and a set of bidirected edges E, we create a 2-XOR-formula ϕ as a conjunction of 2-XOR-clauses using the following reduction. For each edge $e \in E$ connecting vertices (v_i,v_j) we create a 2-XOR-clause C of the form $C = (x_i \oplus x_j)$, where:

1. x_i is b_i if e is positive-incident to vertex v_i and $\neg b_i$ otherwise; and
2. x_j is b_j if e is positive-incident to vertex v_j and $\neg b_j$ otherwise



$$(a \oplus b)$$

$$(\neg c \oplus \neg d)$$

$$(e \oplus \neg f)$$

(a) Bidirected Edge Forms (b) 2-XOR-SAT Clauses

Figure 4.8: (a) There are three forms of bidirected edges: introverted, extraverted, and directed. (b) In the reduction to MAX-2-XORSAT, a clause is created for each bidirected edge in the scaffold graph.

(see Figure 4.8b). The number of steps required for the reduction is linear in the size of E. Figure 4.9 shows the set of weighted 2-XOR-clauses resulting from a reduction of our bidirected graph example.

Each truth assignment of a binary variable b_i in the 2-XOR-formula has a corresponding vertex-orientation assignment of v_i in the bidirected graph: *true* $\equiv$ *forward-orientation*; *false* $\equiv$ *reverse-orientation*. By definition, the truth assignment of b_i will always be opposite that of $\neg b_i$, which is equivalent to saying that the vertex-orientation assignment v_i^+ will always be opposite that of v_i^-. Thus a truth assignment of a negated variable $\neg b_i$ by default assigns the opposite value to the literal b_i.

It remains to be shown then that a truth assignment I which satisfies at least k clauses has a corresponding vertex-orientation assignment A resulting in a subgraph containing at least k directed edges. Such a correspondence can be shown if we can demonstrate that given a truth assignment I to b_i and b_j which satisfies the clause $C = (x_i \oplus x_j)$ representing edge e, the corresponding vertex-orientation assignment A of v_i and v_j will always render e a directed edge. We proceed by considering all cases in which C is satisfied. It is assumed that I initially assigns *true* to all variables in the domain of ϕ and correspondingly A assigns all vertices to *forward-orientation*.

$$\begin{array}{cc}
(a \oplus \neg b, 7) & (d \oplus \neg e, 15) \\
(\neg a \oplus \neg d, 5) & (d \oplus \neg f, 6) \\
(b \oplus \neg c, 8) & (\neg e \oplus f, 6) \\
(b \oplus d, 9) & (\neg e \oplus \neg g, 9) \\
(b \oplus e, 7) & (f \oplus \neg g, 11) \\
(c \oplus e, 5) &
\end{array}$$

Figure 4.9: An example of a set of weighted 2-XOR-clauses resulting from the reduction of the weighted bidirected graph in Figure 4.7. A weighted 2-XOR-clause is of the form $C = (x_i \oplus x_j$, $u)$, where u is the weight for the clause. The 2-XOR-formula formed by the conjunction of these weighted clauses represents an instance of the weighted MAX-2-XORSAT problem.

In the cases where *neither* literal x_i nor x_j is a negated variable (meaning that in the represented edge e, both endpoints are positive-incident) or *both* literals are negated variables (both endpoints are negative-incident), then to satisfy C, I must assign *true* to one variable and *false* to the other. To whichever variable I assigns *false*, the corresponding vertex is assigned the reverse-orientation. This flips one of the two positive-incident endpoints of e and renders e directed. In the remaining case where only one of the two literals is a negated variable (meaning that e is already directed), I must assign *true* to both variables (e does not change) or *false* to both variables (both endpoints in e are flipped, making e a directed edge in the opposite direction) to satisfy C. Thus any truth assignment I which satisfies a clause C renders the corresponding edge e a directed edge. A truth assignment I, therefore, which satisfies at least k clauses in ϕ has a corresponding vertex-orientation assignment A of G resulting in a subgraph containing at least k directed edges.

We have thus shown that any MAX-DIR-SUBGRAPH problem has a linear-time reduction to a MAX-2-XORSAT. To show equivalency, we must now prove that any MAX-2-XORSAT problem has a linear-time reduction to a MAX-DIR-SUBGRAPH problem. Given a 2-XOR-formula ϕ, we create a bidirected graph G with orientable vertices. We create a vertex v_i in G for each variable b_i

in the domain of ϕ. Then for each clause $C = (x_i \oplus x_j)$ in ϕ we create a bidirected edge e linking v_i and v_j where:

1. e is positive-incident to vertex v_i if x_i is not a negated variable and negative-incident to v_i otherwise; and
2. e is positive-incident to vertex v_j if x_j is not a negated variable and negative-incident to v_j otherwise.

The reduction is linear in the sum of the number of variables and the number of clauses in ϕ.

Each vertex-orientation assignment of G has a corresponding truth assignment of ϕ as defined above. We now show that a vertex-orientation assignment A of G which results in a subgraph containing at least k directed edges has a corresponding truth assignment I of ϕ that satisfies at least k clauses. We again do this by showing a one-to-one correspondence between a directed edge in G given the vertex-orientation assignment A and a satisfied clause in ϕ given the corresponding truth assignment I. We proceed by considering all cases in which $A(G)$ renders an edge e linking vertices v_i and v_j a directed edge. It is assumed that A initially assigns all vertices to *forward-orientation* and correspondingly I assigns *true* to all variables in the domain of ϕ.

In the case where e is either introverted (meaning neither variable in the corresponding clause C is negated) or extraverted (both are negated), A must assign *forward-orientation* to either v_i or v_j and *reverse-orientation* to the remaining vertex in order to render e directed. To whichever vertex is assigned *reverse-orientation*, the corresponding variable is assigned *false*, thereby satisfying C. In the remaining case where e is already directed (meaning that one of the two variables in C is a negated variable), then to render e directed, A must assign both v_i and v_j to *forward-orientation* (meaning both variables are *true*). Thus any vertex-orientation assignment A which renders e a directed edge has a corresponding truth assignment I which satisfies the clause C represented by e. Therefore a vertex-orientation assignment A of G which results in a subgraph containing at least k directed edges has a corresponding truth assignment I which satisfies at least k clauses in ϕ.

4.2.4 Equivalence to MAX-CUT

Ultimately our goal is to show that the MAX-DIR-SUBGRAPH problem is equivalent to the MAX-CUT problem, which aside from allowing us to apply any of the several heuristic solutions that have been developed to solve MAX-CUT to the MAX-DIR-SUBGRAPH problem, also proves that the complexity of MAX-DIR-SUBGRAPH is the same as that of MAX-CUT (i.e., NP-complete). We find that the reduction from MAX-2-XORSAT to MAX-CUT is much easier to understand than the reduction directly from the MAX-DIR-SUBGRAPH problem. The reduction follows similar to the approach taken in the reduction from MAX-2-SAT to MAX-CUT (as demonstrated by Garey et al. [16]) in that for each clause in the satisfiability problem, a number of nodes and edges are added to the corresponding graph for which a MAX-CUT solution is to be found. The specifics of the reductions differ sufficiently to warrant detailing the reduction to and from MAX-2-XORSAT.

The decision problem corresponding to the MAX-CUT problem is stated as follows:

$$\text{MAX-CUT} = \{(M, k) \mid M \text{ is a multigraph with a cut of size at least } l\}$$

where a *multigraph* M=(V,E) is a graph allowing duplicate edges and a *cut* in a graph is a partition of V into two distinct subsets S and T. The size of the cut is the number of edges $e \in E$ which have an endpoint in S and an endpoint in T.

We now give a linear-time reduction from MAX-2-XORSAT to MAX-CUT. Given a 2-XOR-formula ϕ with n variables and m clauses, we construct a multigraph M as follows. For each binary variable b_i in the domain of ϕ we create two vertices v_i and $\neg v_i$ in M and add $2m$ edges between the pair (v_i, $\neg v_i$). Then, for each 2-XOR-clause $C = (x_i \oplus x_j)$ in ϕ, we add an edge to M linking x_i and x_j (see Figure 4.10). The reduction is linear in the sum of the number of variables and the number of clauses in ϕ. Figure 4.11 shows the weighted MAX-CUT problem resulting from a reduction of the 2-XOR-formula presented in Figure 4.9.

To complete the reduction, it must be shown that a cut Z of M of size at least l has a corresponding truth assignment I for ϕ that satisfies at least k clauses. We do this by demonstrating a one-to-one correspondence between an edge that is cut in Z and a 2-XOR-clause that is satisfied

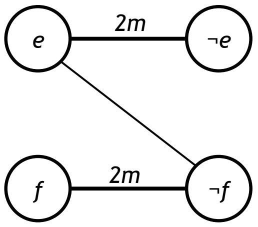


Figure 4.10: The component shown represents the vertices and edges that would be added to the MAX-CUT multigraph for the 2-XOR-SAT clause $(e \oplus \neg f)$.

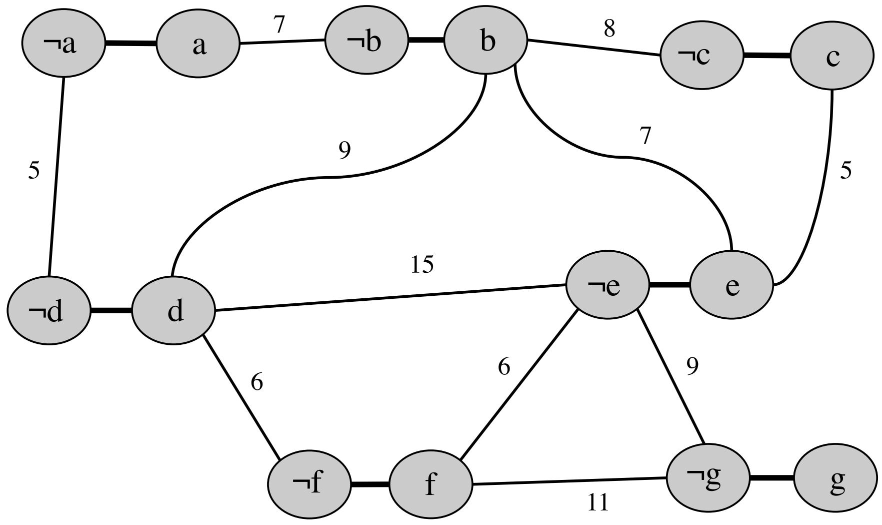


Figure 4.11: An example of a weighted multigraph whose MAX-CUT solution corresponds to the weighted MAX-DIR-SUBGRAPH and MAX-2-XORSAT solutions in Figures 4.7 and 4.9 respectively. Each bold edge linking two vertices v_i and $\neg v_i$ is heavily weighted such as to guarantee its inclusion in the MAX-CUT solution.

by I in ϕ. In doing so, we note that the *complement edge* between the pair (v_i, $\neg v_i$) corresponds to the clause $((x_i \oplus \neg x_i) = 1)$, which is guaranteed by the Complement Law of boolean algebra.

In finding a MAX-CUT solution of M, the partition of a vertex v_i into one of two parts p_0 and p_1 corresponds to the assignment of the corresponding binary variable b_i to 0 or 1 respectively. In reducing from ϕ to M each edge e linking v_i and v_j can be thought of in terms of the 2-XOR-clause $C = (x_i \oplus x_j)$ to which it corresponds: inasmuch as x_i xor x_j must be *true* to satisfy C, v_i xor v_j must be in p_1 in order for e to be cut. Thus finding a maximum cut Z is equivalent to finding a truth assignment I that satisfies all clauses C_i whose corresponding edge e_i is cut in Z. By augmenting the multiplicity of complement edges ($2m$), we guarantee their inclusion in Z, but clauses corresponding to these edges are not explicitly included in ϕ. If the number of edges cut in Z is l and the number of complement edges cut in Z is l_c, then the number of clauses satisfied by the truth assignment I corresponding to Z is $k = l - l_c$.

The linear-time reduction from MAX-CUT to MAX-2-XORSAT is trivial. Given a multigraph M with a set of vertices V and a set of edges E we construct a 2-XOR-formula ϕ as a conjunction of 2-XOR-clauses as follows. For each edge $e \in E$ linking vertices $v_i \in V$ and $v_j \in V$, we create a 2-XOR-clause $C = (x_i \oplus x_j)$. The truth assignment I of each literal x_i in ϕ to 0 or 1 corresponds to the partitioning of the corresponding vertex $v_i \in V$ into one of two partitions p_0 or p_1 respectively. Inasmuch as including C in the MAX-2-XORSAT solution requires either x_i xor x_j to be true, including e in the MAX-CUT solution requires cutting M such that v_i xor v_j is in p_1. In this case, a truth assignment I that satisfies at least k clauses will have a corresponding cut of size at least $l = k$. The reduction is linear in the size of E.

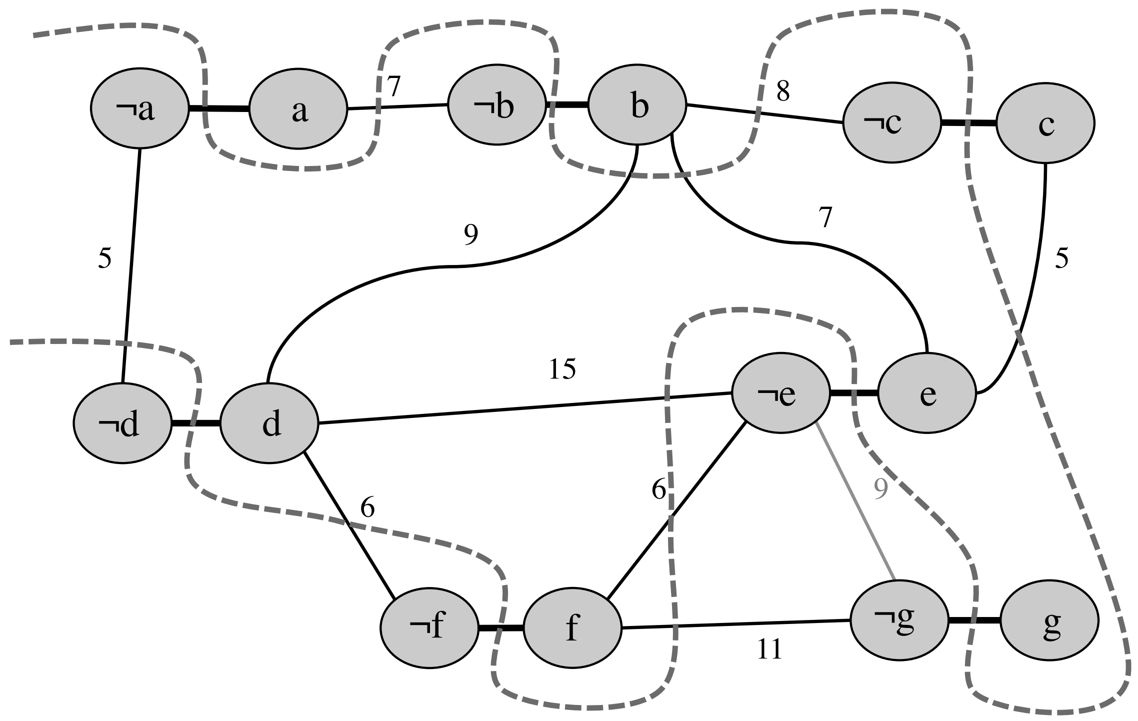
4.2.5 Example Solution

Figure 4.12a shows the optimal solution to the weighted MAX-CUT example problem presented in Figure 4.11. The partition of nodes in the weighted multigraph corresponds directly to the truth-assignment for the weighted 2-XOR-formula referenced in Figure 4.9 (see Figure 4.12b) and the vertex-orientation assignment for the weighted bidirected graph shown in Figure 4.7 (see Figure 4.12c).

4.3 Related Work

Several existing papers discuss the contig orientation problem with varying levels of detail. Pop et al. [38] phrase the contig orientation problem as the problem of coloring a bipartite graph with the colors corresponding to the two possible sequence orientations and implement a heuristic which greedily assigns orientations to contigs and ignores edges which conflict with previous assignments. Donmez et al. [13] explicitly address the issue of contig orientation by making arbitrary orientation assignments which are propagated via a breadth-first-search through the scaffold graph. Li et al. [29], in presenting the SOAPdenovo assembler, mention that following repeat masking, "remaining contigs with compatible connections to each other were linearized" in preparation for scaffold construction, but with limited detail of any specific algorithm. MAIA [35], a graph-based algorithm for integrating multiple assemblies, assigns orientation to contigs via a graph depth-first traversal in order of weight and removes conflicting edges. They use a reference genome to determine start and end nodes. Many other algorithms do not explicitly address the contig orientation problem but rather address the issue in the context of completing a walk through the graph in the scaffold construction phase [4, 7, 49].

Many prevalent assembly algorithms do not use the formal bidirected graph notation when illustrating scaffold graphs; however, the most common notations can be shown to be notationally equivalent to bidirected graph notation. Several papers [35, 38, 49] use a notation similar to the one shown in Figure 4.13, where it is the *nodes* that are depicted with an inherent directionality using some sort of biterminal distinction: one node terminal represents the biological 5' end of the contig sequence (as read from the sequencing machine) and the other terminal represents the 3' end. Edges are then connected between the terminals or *ends* of a node in one of three ways: between 3' and 3' ends; between 5' and 5'; or between 3' and 5' ends (e.g., see Figure 4.13). In this notation, which



(a)

$a = \mathbf{F}$

$b = \mathbf{F}$

$c = \mathbf{F}$

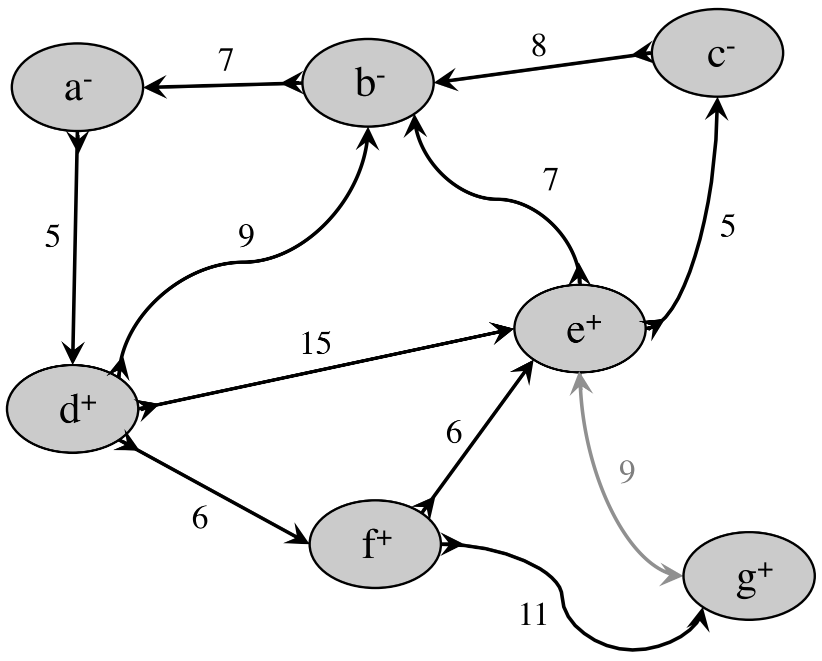
$d = \mathbf{T}$

$e = \mathbf{T}$

$f = \mathbf{T}$

$g = \mathbf{T}$

(b)



(c)

Figure 4.12: (a) Shown is the BiqMac optimal solution to the weighted MAX-CUT example problem presented in Figure 4.11. Note that the edge from $\neg e$ to $\neg g$ is uncut. (b) Shown is the corresponding truth assignment for the weighted 2-XOR-formula referenced in Figure 4.9 which satisfies all the weighted clauses except $(\neg e \oplus \neg g)$. (c) Shown is the corresponding vertex-orientation assignment for the weighted bidirected graph shown in Figure 4.7. Note that as a result of the assignment, all edges are directed except the edge between e^+ and g^+.

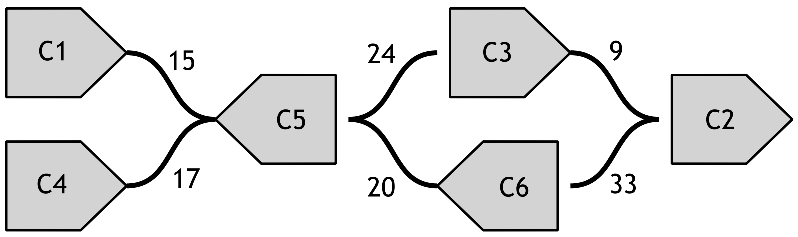


Figure 4.13: *Biterminal Scaffold Graph Notation*. The scaffold graph from Figure 4.5 is here depicted using biterminal notation. Directionality is inherent using a biterminally distinctive node (arrow points from 5' to 3' of forward strand). This notation is often preferred because the spatial arrangement of nodes and edges more closely approximates the biological model.

is easily relatable to the biological system being depicted, a valid reconstruction or *walk* through a node requires entering and exiting the node via opposite terminals.

Formal bidirected graph notation, as first introduced by Edmonds and Johnson [14], is used in several papers to deal with double-strandedness in overlap consensus modeling, a problem related to but very different from scaffolding [20, 30, 34]. Such notation is often represented using an incidence matrix $I : V\ x\ E \mapsto \{-2, -1, 0, 1, 2\}$. If an edge e is not incident to a vertex x then $I(x,e) = 0$; if e is positive-incident to x, then $I(x,e) = 1$; and if e is negative-incident to x, then $I(x,e) =$ -1 [30].

The MAX-2-XORSAT problem has been discussed primarily as a variant of the maximum satisfiability problem in general and in conjunction with the MAX-CUT problem. Daudé et al. [10] consider the MAX-CUT problem on random connected graphs, specifically focusing on the distance from bipartiteness of a graph (i.e. number of edges to be removed to turn it into a bipartite graph), and cursorily observe an equivalence relationship between MAX-2-XORSAT and MAX-CUT. Rasendrahasina et al. [40] randomly select 2-XOR formulae in order to characterize phase transitions of the MAX-2-XORSAT problem, describing how likely a formula is satisfiable as the number of clauses increases.

Several algorithms have been proposed to heuristically or optimally solve MAX-SAT and weighted MAX-SAT problems. Joy et al. [23] describe an integer programming branch-and-cut algorithm which uses a GSAT heuristic. Borchers et al. [6] present a two-phase algorithm that uses an initial heuristic solution to find a provably optimal solution based on the Davis-Putnam-Loveland algorithm. They show that in some cases this approach outperforms the integer programming solution, but is less effective in some other problem classes, including MAX-2-SAT problems.

MAX-CUT, which was one of Karp's original NP-complete problems [24], has been heavily studied and a number of effective approximation algorithms have been proposed. Ding et al. [11] present a min-max cut algorithm based on the min-max clustering principle and demonstrate its effectiveness compared to several other algorithms. Sahni and Gonzales [44] present a linear-time ½-approximation algorithm (meaning the algorithm delivers a solution that is guaranteed to be at least ½ times the optimal value). This method considers vertices in random order and assigns each vertex to whichever partition minimizes the weight of connecting edges within the set.

Goemans and Williamson [17] present what is commonly accepted as the best known approximation (a .878-approximation) by randomly rounding the solution to a nonlinear-programming relaxation. This is the first ever use of semi-definite programming (SDP) for approximation. Khot et al. [25] demonstrate that if the unique games conjecture is true, the Goemans-Williamson algorithm is the best possible approximation algorithm for MAX-CUT. Rendl et al. [42] provide a survey of several popular and recent methods to solving MAX-CUT problems including linear-programming based relaxation, SDP relaxation, convex quadratic equations, and multiple Branch-and-Bound approaches. They present their own Branch-and-Bound method, BiqMac, which utilizes SDP and applies the Goemans-Williamson hyperplane rounding technique [18] in heuristically generating feasible solutions. Their primary contribution is using Lagrangian duality for solving the semi-definite MAX-CUT relaxation.

4.4 Methods

4.4.1 Algorithms

Prior to discovering the reduction to weighted MAX-CUT, we developed a greedy heuristic algorithm to solve the weighted MAX-DIR-SUBGRAPH problem (see Algorithm 1). This algorithm follows similarly to Kruskal's algorithm for finding a minimum spanning tree [26] in that we start by making each vertex in the graph its own tree and then add edges which combine distinct trees to form larger trees. In combining trees t_i and t_j via edge e, we flip vertex orientations for all vertices in t_j as necessary so that e is always directed. Edges are considered in order of decreasing weight. Any edge e linking vertices v_i and v_j within the same tree t_i is added iff, given the current vertex-orientation assignment of v_i and v_j in t, e is a directed edge. We ensure that the final subgraph contains solely directed edges by only adding directed edges and ensuring that directed edges remain directed as a result of vertex orientation changes.

Algorithm 1 WEIGHTED MAX-DIR-SUBGRAPH GREEDY HEURISTIC

```
Input: A weighted bidirected graph, $G$
 1: Create a forest, $F$, with a tree $t_i$ for each vertex $v_i$ in $G$
 2: Create a set $S$ containing all the edges in $G$
 3: while $S$ is not empty do
 4:     Remove an edge $e$ with maximum weight from S
 5:     if $e$ connects two different trees, $t_1$ and $t_2$, then
 6:         add $e$ to F, combining $t_1$ and $t_2$ into one tree
 7:         if $e$ is not a directed edge then
 8:             for all vertices $v_2$ in $t_2$ do
 9:                 Flip vertex-orientation assignment of $v_2$
10:             end for
11:         end if
12:     else if $e$ is a directed edge then
13:         add $e$ to $F$
14:     else
15:         discard $e$
16:     end if
17: end while
18: return $F$, a weighted directed subgraph
```

The relative performance of four solutions to the contig orientation problem, including our greedy heuristic, were measured:

1. *Random* - a simple ½-approximation algorithm which considers edges in a random order and assigns an orientation to adjacent contigs that is consistent with previous orientation assignments [44].
2. *Greedy* - our aforementioned greedy heuristic solution, which considers edges in order of decreasing weight and assigns an orientation to adjacent contigs that is consistent with previous orientation assignments (see Algorithm 1).
3. *BiqMac* - a Branch-and-Bound heuristic algorithm for solving weighted MAX-CUT problems which uses SDP relaxation and a relative bound precision criterion [42].
4. *Optimal* - the optimal solution (where calculable by BiqMac), which was found using SDP relaxations strengthened by triangle inequalities [42].

4.4.2 Datasets

The algorithms were assessed on three bidirected graphs: two synthetic genome scaffold graphs, and a scaffold graph from the raspberry genome. We describe each of these graphs in more detail.

Synthetic Genome (w/o Errors)

A 1.25-Mb genome with haplotype variation (heterozygosity rate ≈4.0%) was synthesized from chromosome 25 of the zebra finch (ACCN NC_011489.1) using an in-house software package called HapMaker [36]. The following were generated for contig assembly: a set of 250bp single reads at 80x coverage; a 4kb paired-end library at 8x coverage (250bp reads); and a 20kb paired-end library at 8x coverage (250bp reads). Newbler 2.6 was used to assemble contigs with the following parameters: -mi 98 -minlen 30 -infoall -ml 50 -a 1 -nohet -large. Reads were mapped and a scaffold graph formed using ScaffoldScaffolder [5] with Bowtie v0.12.9 [28]. Only the 4kb libraries were used for the scaffold graph construction. ScaffoldScaffolder was run with the parameters -minalign

30 -algorithm greedy -minsupport 1 –showExcludedEdges -ploidy 2. Parameters for Bowtie were -v 2 -a -m 1 -f.

Synthetic Genome (w/ Errors)

Using the 1.25-Mb genome reference, a paired-read library was generated from ART v1.3.1 [19] using a quality profile with an estimated error rate of 3.35%. ART parameters were -l 75 -f 80 -qs 0 -s 10 -m 200. Contig assembly was performed using Newbler, and a scaffold graph was created using ScaffoldScaffolder and Bowtie2 v2.0.6 (which allows for more mismatches than Bowtie) [27]. Newbler was run with parameters -mi 90 -minlen 30 -infoall -ml 50 -a 1 -nohet -large. ScaffoldScaffolder was run with parameters -minalign 30 -algorithm greedy -minsupport 1 –showExcludedEdges -ploidy 2. Bowtie2 parameters were –end-to-end -f -k 1 –score-min L,-0.6,-1.2.

Raspberry Genome

We assessed performance on a scaffold graph constructed from contigs assembled using Newbler on reads from a *Rubus idaeus cultivar heritage* raspberry genome (350 Mb). Contigs for the raspberry genome were assembled using reads from a combination of Illumina HiSeq and 454 sequencing technologies. Single reads and short insert Illumina paired reads were sampled at high coverage and paired 454 reads (with insert sizes ranging from 3kb to 20kb) were sampled at low coverage. Due to the size of the graph, only contigs greater than 500 bp in length were considered in the graph. A scaffold graph was generated using ScaffoldScaffolder and Bowtie2. Only Illumina reads were used for scaffolding inference. Newbler was run with parameters -mi 90 -minlen 30 -infoall -ml 50 -a 1 -nohet -large. ScaffoldScaffolder was run with parameters -minalign 30 -algorithm greedy -minsupport 1 –showExcludedEdges -ploidy 2. Bowtie2 parameters were –end-to-end -f -k 1 –score-min L,-0.6,-1.2.

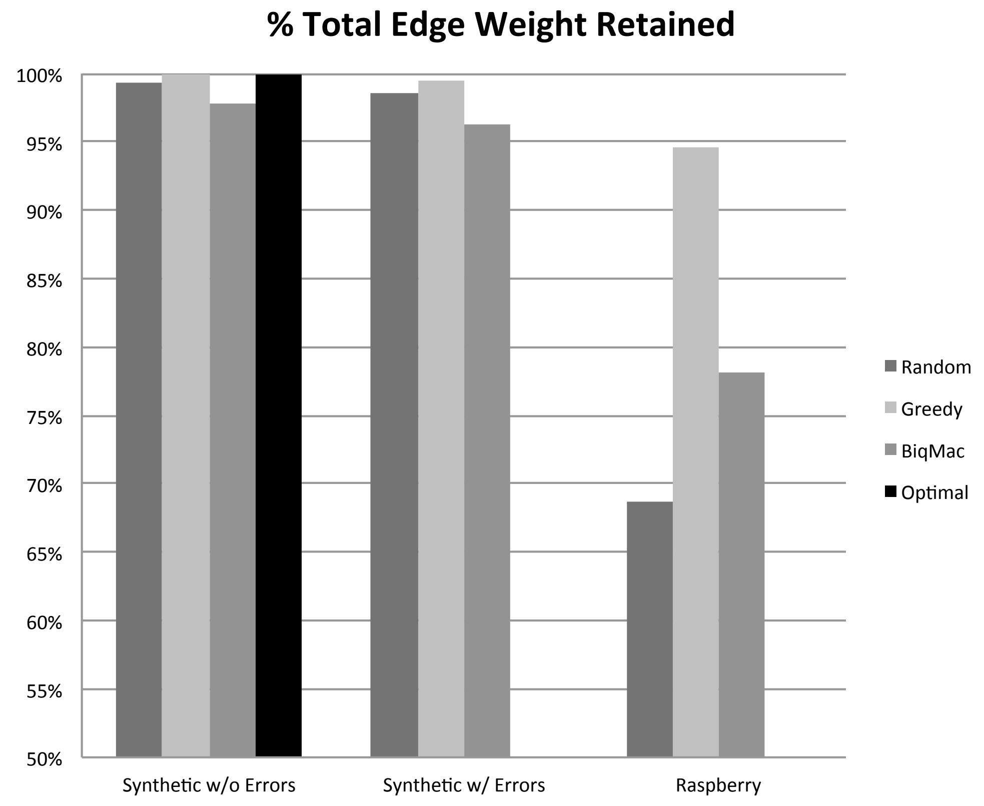


Figure 4.14: *Comparative performance of weighted MAX-DIR-SUBGRAPH solutions*. The optimal solution is shown where it was computable. In all three scaffold graphs, the greedy algorithm retained the most total edge support.

4.4.3 Metrics

Solutions were assessed by obtaining the following four metrics: the total count of edges retained; the total weight of edges retained; the total count of edges excluded; and the total weight of edges excluded. Optimal solutions for the synthetic genome (w/ errors) and the raspberry genome were not obtainable in a reasonable amount of time due to the size of these graphs.

4.5 Results

The results of the experiment are shown in Table 4.1. The percent of total edge weight retained by each algorithm on each of the datasets is summarized in Figure 4.14.

Table 4.1: Comparative Results on the Contig Orientation Problem

	Random	Greedy	BiqMac	Optimal
SYNTHETIC GENOME (W/O ERRORS) RESULTS				
Retained Edges				
Count	790	*791*	748	*791*
Weight	6553	*6588*	6454	*6588*
Excluded Edges				
Count	3	*2*	45	*2*
Weight	44	*9*	143	*9*
SYNTHETIC GENOME (W/ ERRORS) RESULTS				
Retained Edges				
Count	1,695	1,689	1,543	*n/a*
Weight	39,172	39,507	38,226	*n/a*
Excluded Edges				
Count	19	25	171	*n/a*
Weight	560	225	1,506	*n/a*
RASPBERRY GENOME RESULTS				
Retained Edges				
Count	528,616	*642,834*	559,199	*n/a*
Weight	6,604,235	*9,080,859*	7,501,716	*n/a*
Excluded Edges				
Count	329,925	*215,707*	299,342	*n/a*
Weight	3,002,050	*525,426*	2,104,569	*n/a*

In the synthetic genome graphs, the random and greedy algorithms outperformed the SDP solution, with the greedy heuristic retaining the most in total edges and total edge weight. We suspect that the superior performance of the random and greedy algorithms may be due to the fact that in scaffold graphs, the edges derive from a molecular model that predefines an orientation assignment for all sequences. Consequently we would expect that the vast majority of the supportive evidence would be internally consistent; relatively few spurious edges require exclusion. In addition, a scaffold graph is quite sparse (i.e., average in- and out-edge degrees are between 1 and 2) and linear by nature, thus rendering the problem much simpler. We suspect that in such a graph, a local optimum will often be part of the global optimum.

The raspberry genome graph, however, is quite different from the synthetic graphs: the greedy algorithm retained $\approx$15% more edge support than the SDP algorithm and $\approx$25% more support than the random algorithm. One possible explanation for the inferior performance of the random heuristic algorithm is that inclusion of a spurious edge (via random selection) is more costly in the raspberry genome than in the synthetic genomes. This is because the average supportive evidence for edges in the raspberry genome is nearly five times more than the average supportive evidence for edges in either of the synthetic genomes, which comes as an artifact of the biological sampling. Although the weight of a spurious edge in any scaffold graph is about the same (e.g., less than 5), the weight of a non-spurious edge in the raspberry genome is far greater than a non-spurious edge in one of the synthetic genomes. This might explain why the random algorithm retains far less edge support than the greedy algorithm in the raspberry scaffold graph.

The superior performance of the greedy heuristic algorithm with respect to the SDP solution is likely due to this key observation about the nature of scaffold graphs: not only do spurious, lowly-weighted edges contribute less to total retained edge support than heavily-weighted edges; they are also unlikely to belong in the optimal solution. Conversely, heavily-weighted retained edges not only contribute more to total edge support than lowly-weighted edges; they are more likely to be part of the optimal solution. The greedy algorithm thus maximally favors edges which are likely to belong in the optimal solution, which may help to explain why it retained far more edge weight than even the SDP solution in the raspberry scaffold graph.

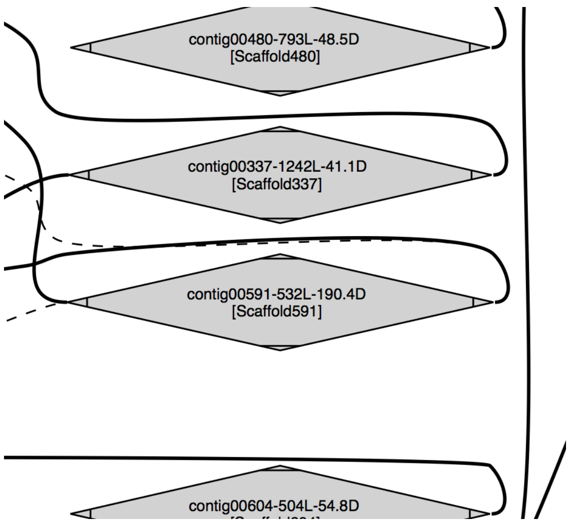
4.6 Discussion

We noted a peculiarity when examining the subgraph produced using our greedy heuristic algorithm on the synthetic genome without errors: the two excluded edges were both adjacent to contig 591 (see Figure 4.15a). In addition we noted that contig 591 has an average *sequence depth* (a value indicative of the number of nucleotides contributing to the assembly at a given locus) of 190.4. This is almost exactly twice the normal diploid depth, making it a likely candidate for being a two-copy repeat (see Figure 4.15b). We used BLAST [1] to find where contig 591 aligned to the known reference sequence and discovered that not only did it align perfectly at two locations, but that the two matching sequences were inversions (see Figure 4.15c).

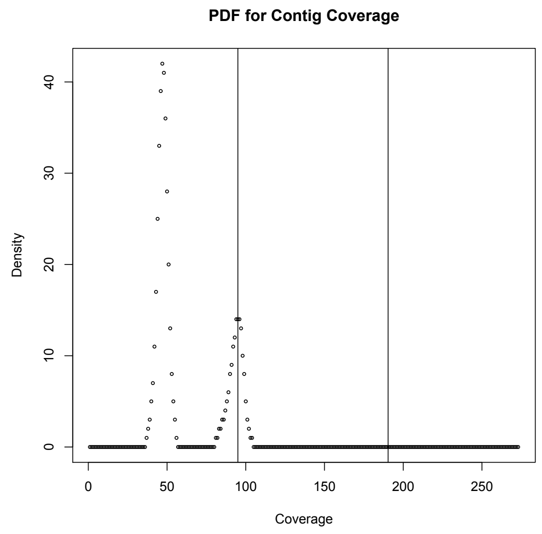
This case illustrates that if repeats are not screened, they can present exceptions to the contig-orientation problem, and more specifically to the single-orientation assumption. The single-orientation assumption will hold true if the contig represents a sequence which repeats in the same orientation in a scaffold (e.g., tandem repeats). However, as our example illustrates, repeats can also exist in differing orientations.

Inverted repeats (also called *palindromic sequences*), like contig 591, represent identical sequences from two distinct places in a scaffold, where the orientations of the two sequences are opposite one another (see Figure 4.16a). Among their several biological roles, inverted repeats are used to detect the boundaries of transposons (e.g., [43]) and are instrumental in transcriptional regulation (e.g., [33]). In essence, assigning a single orientation to such a contig prevents a viable scaffold reconstruction from occurring, prematurely fragmenting the assembly. This scenario can be easily resolved prior to assigning contig-orientations either by special handling or screening of repeat contigs, as done in [29, 38].

This example also serves to illustrate a significant and unanticipated ramification of the contig orientation problem: whereas the contig orientation assignment is designed to remove



(a)



(b)

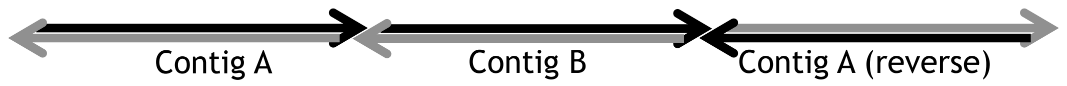
```
qseqid     sseqid     pident     length     mismatch gapopen     qstart     qend       sstart     send evalue

contig591  Subject_1  100.00     532        0          0           1          532        737796     737265  0.0
contig591  Subject_1  100.00     532        0          0           1          532        741567     742098  0.0
```

(c)

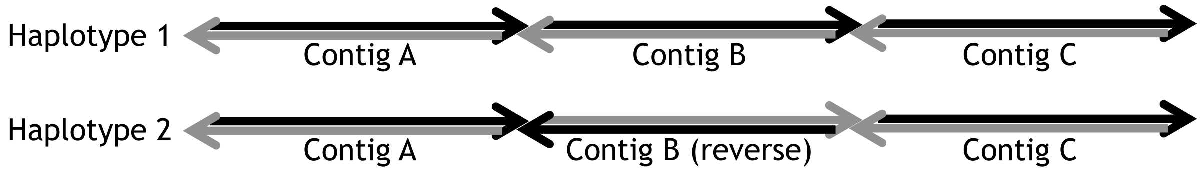
Figure 4.15: (a) In this closeup of the subgraph produced using the greedy heuristic algorithm on a synthetic genome without errors, we see that the only two excluded edges (dotted lines) were both adjacent to contig 591. (b) A probability density function of the sequence depths of all contigs in the contig set shows a diploid depth of about 95 (left vertical line). The 190.4 depth of contig 591 (right vertical line) is almost exactly twice the diploid coverage, suggestive of a repeat contig. (c) The BLAST result of contig 591 to the reference verified that the contig was not only repetitive, but also an inversion.

erroneous linkages from a graph, in cases where such linkages are removed via other means (e.g., minimum support threshold, next-generation error-correction), the contig orientation solution serves to identify (via exclusion) biologically viable exceptions to the single-orientation assumption.

Inverted repeats are not the only scenario which would violate the single-orientation assumption. One other such scenario is the case of inverted haplotypes. Most large genomes exist as *diploid* or *polyploid*, meaning that rather than there being a single cellular genome to assemble, there are two or more genomic versions in each cell. These versions are referred to as *haplotypes*. An *inverted haplotype* is a sequence which is identical but oppositely-oriented at corresponding locations on analogous chromosomes (see Figure 4.16b). Such inversions are often biologically



(a) Inverted Repeat

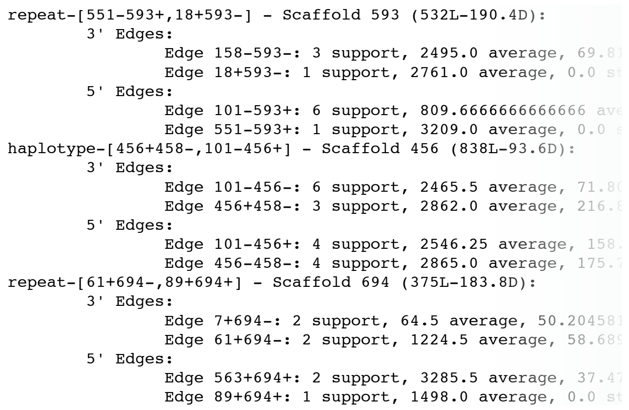


(b) Inverted Haplotypes

Figure 4.16: *Violations of the Contig Orientation Problem Assumption.* (a) In an inverted repeat, a sequence (e.g., contig A) is included in the reconstruction twice in opposite orientations. (b) In a polyploid genome, an inverted haplotype is a case where a sequence (e.g., contig B) is included in opposite orientations on different haplotypes.

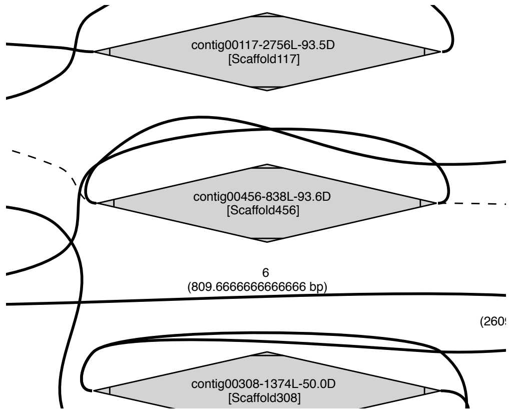
significant and have been specifically shown to be associated at times with mental retardation, microdeletion syndrome, renal cysts and diabetes (RCAD) syndrome, epilepsy, schizophrenia, and autism [2, 51]. Most assembly algorithms have not been specifically designed for diploid genome assembly and assume that where multiple haplotypes do exist, they can be readily merged to form a single "reference" sequence. In doing so, inverted haplotype differences are metaphorically "swept under the rug," which is perhaps why biologists have lamented that "unlike other types of structural variation, little is known about inversion variants within normal individuals because such events are typically balanced and are difficult to detect and analyze by standard molecular approaches" [2] (see also [3]).

We hypothesized that just as a contig orientation solution was able to identify an inverted repeat, it may also be able to identify inverted haplotypes. To test this theory, we developed a module in ScaffoldScaffolder to automatically generate a detailed report (and corresponding FASTA file) of potential inverted repeats and inverted haplotypes. In the module, candidates are internally identified as any contig (1) having at least 2 connecting edges from both ends and (2) which is connected to 2 or more excluded edges. The candidates are classified as inverted repeats or inverted haplotypes based on the location of each candidate contig in a probability density function of contig



```
repeat-[551-593+,18+593-] - Scaffold 593 (532L-190.4D):
        3' Edges:
                Edge 158-593-: 3 support, 2495.0 average, 69.8
                Edge 18+593-: 1 support, 2761.0 average, 0.0 s
        5' Edges:
                Edge 101-593+: 6 support, 809.6666666666666 ave
                Edge 551-593+: 1 support, 3209.0 average, 0.0 s
haplotype-[456+458-,101-456+] - Scaffold 456 (838L-93.6D):
        3' Edges:
                Edge 101-456-: 6 support, 2465.5 average, 71.8
                Edge 456+458-: 3 support, 2862.0 average, 216.
        5' Edges:
                Edge 101-456+: 4 support, 2546.25 average, 158.
                Edge 456-458-: 4 support, 2865.0 average, 175.
repeat-[61+694-,89+694+] - Scaffold 694 (375L-183.8D):
        3' Edges:
                Edge 7+694-: 2 support, 64.5 average, 50.20458
                Edge 61+694-: 2 support, 1224.5 average, 58.68
        5' Edges:
                Edge 563+694+: 2 support, 3285.5 average, 37.4
                Edge 89+694+: 1 support, 1498.0 average, 0.0 s
```

(a)



(b)

```
qseqid          sseqid    pident    length    mismatch gapopen    qstart    qend    sstart    send   eval

contig593-rep   Ref       100.00    532       0        0          1         532      737796   737265  0.0
contig593-rep   Ref       100.00    532       0        0          1         532      741567   742098  0.0
contig593-rep   Var       99.62     532       2        0          1         532      741567   742098  0.0
contig593-rep   Var       99.25     532       4        0          1         532      737796   737265  0.0
contig456-hap   Ref       100.00    838       0        0          1         838      743297   744134  0.0
contig456-hap   Var       99.76     838       2        0          1         838      744134   743297  0.0
contig694-rep   Ref       100.00    375       0        0          1         375      811544   811170  0.0
contig694-rep   Ref       99.47     376       1        1          1         375      816180   816555  0.0
contig694-rep   Var       100.00    375       0        0          1         375      811544   811170  0.0
contig694-rep   Var       99.20     376       2        1          1         375      816180   816555  0.0
```

(c)

Figure 4.17: (a) The inversion report shows contig 456 whose edges and depth distinguish it as a probable inverted haplotype. (b) A graphic representation of the subgraph at contig 456 showing two excluded edges. (c) The BLAST result of all three inversions from the inversion report, verifying their correct identification and classification.

coverage (e.g., Figure 4.15b). Inverted haplotype candidates also require that each of two adjacent contigs be linked via edges from both ends of the candidate.

We tested our new predictive module using a synthesized diploid genome (heterozygosity rate ≈0.2%) from the zebra finch chromosome 25 containing an inverted haplotype. We generated error-free reads for assembly with Newbler and a graph was created using ScaffoldScaffolder and Bowtie. Using our greedy heuristic algorithm, we assigned orientations to the contigs which resulted in a subgraph which excluded 12 edges. The potential inversion report listed two inverted repeats and an inverted haplotype (see Figure 4.17a). We manually verified that all three inversions were accurate using BLAST and verified that the inverted haplotype aligned at the expected location in the reference haplotype genomes (see Figure 4.17c).

Significantly more testing and research is required to assess the efficacy of this method on graphs of varying sizes and complexities, with inverted elements of varying lengths, and in the presence of errors; however, both the theory and our small test confirm that edges which are excluded in solving the contig orientation problem can be used to detect inverted repeats and inverted haplotypes in *de novo* assemblies, particularly when such sequences are adjacent to multiple or heavily supported excluded edges.

4.7 Conclusion

The contig orientation problem, which we have formally framed as the MAX-DIR-SUBGRAPH problem, has been heretofore treated only cursorily in the literature. While several algorithms make reference to the problem, few (e.g., [38]) explicitly acknowledge it as an NP-complete optimization problem, a fact which we have formally proven. The problem has traditionally been addressed (somewhat apologetically) using a greedy heuristic algorithm, similar to that which we have herein presented. We have set forth a linear-time reduction from the MAX-DIR-SUBGRAPH problem to the MAX-CUT problem and have assessed the relative performance of the traditional greedy approach with a more advanced MAX-CUT heuristic solution and with a random heuristic solution. Our results suggest that the greedy heuristic algorithm outperforms other algorithms due to the nature of scaffold graphs. In such graphs, heavier-weighted edges are inherently pre-disposed towards being included in the optimal solution, and thus a greedy algorithm, which maximally favors such edges, approximates an ideal solution.

One unanticipated outcome of this study has been the discovery of a novel method for identifying inverted repeats and inversion variants, both of which contradict the basic single-orientation assumption. Such inversions have previously been noted as being difficult to detect and are directly involved in the genetic mechanisms of several diseases. Thus this method, which we have implemented as a module of ScaffoldScaffolder, has the potential to assist in the automated discovery of biologically significant features in *de novo* genome assembly.

Chapter 5

Overview

5.1 Contributions

In considering the problem of heterozygous diploid genome assembly, we have focused primarily on what is termed the *contig orientation problem*. We have been able to formally define this problem using bidirected graph theory, and have shown it to be equivalent to existing graph decision problems (weighted MAX-2-XORSAT and weighted MAX-CUT). The problem has traditionally been solved (somewhat apologetically) using a greedy heuristic algorithm, similar to that which we have herein presented. We have set forth a linear-time reduction from the MAX-DIR-SUBGRAPH problem to the MAX-CUT problem and have assessed the relative performance of the traditional greedy approach with a more advanced MAX-CUT heuristic solution and with a random heuristic solution. Our results suggest that the greedy heuristic algorithm actually outperforms other algorithms due to the nature of scaffold graphs. In such graphs, heavier-weighted edges are inherently pre-disposed towards being included in the optimal solution, and thus a greedy algorithm, which maximally favors such edges, approximates an ideal solution.

We have demonstrated how this approach to the contig orientation problem facilitates *de novo* feature extraction in diploid genomes, specifically inverted sequences. This illustrates that adaptation of existing graph algorithms to the problem of diploid genome assembly does aid in the resolution of haplotype-specific variation. Furthermore, we have created ScaffoldScaffolder, a highly-configurable standalone heterozygous genomic scaffolder, which includes a module for this diploid feature extraction.

5.2 Future Work

The contig orientation problem is preliminary to the task of scaffold formation. Future work focuses on the creation of linear, haplotype-specific scaffolds via further edge reduction in the scaffold graph. An algorithm has been developed and implemented which uses one of several linear classifiers (e.g., multilayer perceptron, support vector machine, etc.) to classify pairs of contigs as homologous or non-homologous. The classifier is trained on characteristics of homologous sequences as found in what are termed "bubble structures", which tend to accurately indicate homologous, heterozygous genomic regions. Feature values are based on sequence depths, lengths, and similarities. Once homologous contigs are identified, haplotype-specific scaffolds can be formed from each homolog. An important follow-up will then be the *phasing* of haplotypes, which refers to the process of correctly scaffolding the correct combination of haplotype-specific variants in each homologous linear scaffold. This novel approach to heterozygous genomic scaffolding, in tandem with the research already completed, promises to provide leading solutions in diploid genome assembly.

References

[1] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[2] Francesca Antonacci, Jeffrey M Kidd, Tomas Marques-Bonet, Mario Ventura, Priscillia Siswara, Zhaoshi Jiang, and Evan E Eichler. Characterization of six human disease-associated inversion polymorphisms. *Human molecular genetics*, 18(14):2555–2566, 2009.

[3] Vikas Bansal, Ali Bashir, and Vineet Bafna. Evidence for large inversion polymorphisms in the human genome from HapMap data. *Genome research*, 17(2):219–230, 2007.

[4] S. Batzoglou, D.B. Jaffe, K. Stanley, J. Butler, S. Gnerre, E. Mauceli, B. Berger, J.P. Mesirov, and E.S. Lander. Arachne: a whole-genome shotgun assembler. *Genome research*, 12(1): 177–189, 2002.

[5] P Bodily, J Price, M Clement, and Q Snell. ScaffoldScaffolder: An aggressive scaffold finishing algorithm. In *Proceedings of the 2012 International Conference on Bioinformatics & Computational Biology*, pages 385–390. CSREA Press, 2012.

[6] Brian Borchers and Judith Furman. A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems. *Journal of Combinatorial Optimization*, 2(4):299–306, 1998.

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

[8] George M Church, Yuan Gao, and Sriram Kosuri. Next-generation digital information storage in DNA. *Science*, 337(6102):1628–1628, 2012.

[9] N.L. Clement, Q. Snell, M.J. Clement, P.C. Hollenhorst, J. Purwar, B.J. Graves, B.R. Cairns, and W.E. Johnson. The GNUMAP algorithm: Unbiased probabilistic mapping of oligonucleotides from next-generation sequencing. *Bioinformatics*, 26(1):38–45, 2010.

[10] Hervé Daudé, Conrado Martínez, Vonjy Rasendrahasina, and Vlady Ravelomanana. The MAX-CUT of sparse random graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 265–271. SIAM, 2012.

[11] Chris HQ Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 107–114. IEEE, 2001.

[12] Nilgun Donmez and Michael Brudno. Hapsembler: an assembler for highly polymorphic genomes. In *Research in Computational Molecular Biology*, pages 38–52. Springer, 2011.

[13] Nilgun Donmez and Michael Brudno. Scarpa: scaffolding reads with practical algorithms. *Bioinformatics*, 29(4):428–434, 2013.

[14] Jack Edmonds and Ellis L Johnson. Matching: A well-solved class of integer linear programs. In *Combinatorial structures and their applications (Gordon and Breach*. Citeseer, 1970.

[15] D. Fasulo, A. Halpern, I. Dew, and C. Mobarry. Efficiently detecting polymorphisms during the fragment assembly process. *Bioinformatics*, 18(suppl 1):S294–S302, 2002.

[16] Michael R Garey, David S. Johnson, and Larry Stockmeyer. Some simplified *NP*-complete graph problems. *Theoretical computer science*, 1(3):237–267, 1976.

[17] Michel X Goemans and David P Williamson. .879-approximation algorithms for MAX CUT and MAX 2SAT. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 422–431. ACM, 1994.

[18] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[19] Weichun Huang, Leping Li, Jason R Myers, and Gabor T Marth. ART: A next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594, 2012.

[20] Benjamin G Jackson and Srinivas Aluru. Parallel construction of bidirected string graphs for genome assembly. In *37th International Conference on Parallel Processing, 2008. ICPP'08*, pages 346–353. IEEE, 2008.

[21] D.B. Jaffe, J. Butler, S. Gnerre, E. Mauceli, K. Lindblad-Toh, J.P. Mesirov, M.C. Zody, and E.S. Lander. Whole-genome sequence assembly for mammalian genomes: Arachne 2. *Genome research*, 13(1):91–96, 2003.

[22] T. Jones, N.A. Federspiel, H. Chibana, J. Dungan, S. Kalman, BB Magee, G. Newport, Y.R. Thorstenson, N. Agabian, PT Magee, et al. The diploid genome sequence of *Candida albicans*. *Proceedings of the National Academy of Sciences of the United States of America*, 101(19): 7329, 2004.

[23] Steve Joy, John Mitchell, and Brian Borchers. A branch and cut algorithm for MAX-SAT and weighted MAX-SAT. In *Satisfiability problem: Theory and Applications, volume 35 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science*. Citeseer, 1997.

[24] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[25] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1): 319–357, 2007.

[26] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[27] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature Methods*, 9(4):357–359, 2012.

[28] Ben Langmead, Cole Trapnell, Mihai Pop, Steven L Salzberg, et al. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*, 10(3):R25, 2009.

[29] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, et al. De novo assembly of human genomes with massively parallel short read sequencing. *Genome research*, 20(2):265–272, 2010.

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

[31] J.R. Miller, A.L. Delcher, S. Koren, E. Venter, B.P. Walenz, A. Brownley, J. Johnson, K. Li, C. Mobarry, and G. Sutton. Aggressive assembly of pyrosequencing reads with mates. *Bioinformatics*, 24(24):2818, 2008.

[32] Sayyed R Mousavi, Maryam Mirabolghasemi, Nadia Bargesteh, and Majid Talebi. Effective haplotype assembly via maximum boolean satisfiability. *Biochemical and Biophysical Research Communications*, 404(2):593–598, 2011.

[33] Marieelle W. M. Muskens, Adrieenne P. A. Vissers, Joseph N. M. Mol, and Jan M. Kooter. Role of inverted DNA repeats in transcriptional and post-transcriptional gene silencing. *Plant Molecular Biology*, 43(2-3):243–260, 2000. URL `http://search.proquest.com/docview/864953529?accountid=4488`.

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

[35] Jurgen Nijkamp, Wynand Winterbach, Marcel van den Broek, Jean-Marc Daran, Marcel Reinders, and Dick de Ridder. Integrating genome assemblies with MAIA. *Bioinformatics*, 26 (18):i433–i439, 2010.

[36] Nozomu Okuda. HapMaker: Synthetic haplotype generator. *preprint*, 2013. URL `http://dna.cs.byu.edu/presentations/`.

[37] M. Pop. Genome assembly reborn: Recent computational challenges. *Briefings in bioinformatics*, 10(4):354–366, 2009.

[38] M. Pop, D.S. Kosack, and S.L. Salzberg. Hierarchical scaffolding with Bambus. *Genome Research*, 14(1):149–159, 2004.

[39] Jared C Price, Joshua A Udall, Paul M Bodily, Judson A Ward, Michael C Schatz, Justin T Page, James D Jensen, Quinn O Snell, and Mark J Clement. De novo identification of "heterotigs" towards accurate and in-phase assembly of complex plant genomes. In *Proceedings of the 2012 International Conference on Bioinformatics & Computational Biology*, pages 144–150. CSREA Press, 2012.

[40] Vonjy Rasendrahasina and Vlady Ravelomanana. Limit theorems for random MAX-2-XORSAT. In Alejandro López-Ortiz, editor, *LATIN 2010: Theoretical Informatics*, volume 6034 of *Lecture Notes in Computer Science*, pages 320–331. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12199-9. doi: 10.1007/978-3-642-12200-2_29. URL `http://dx.doi.org/10.1007/978-3-642-12200-2_29`.

[41] J.A. Reinhardt, D.A. Baltrus, M.T. Nishimura, W.R. Jeck, C.D. Jones, and J.L. Dangl. De novo assembly using low-coverage short read sequence data from the rice pathogen *Pseudomonas syringae* pv. *oryzae*. *Genome research*, 19(2):294–305, 2009.

[42] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2): 307–335, 2010.

[43] Donald C Rio and Gerald M Rubin. Identification and purification of a *Drosophila* protein that binds to the terminal 31-base-pair inverted repeats of the P transposable element. *Proceedings of the National Academy of Sciences*, 85(23):8929–8933, 1988.

[44] Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *Journal of the ACM (JACM)*, 23(3):555–565, 1976.

[45] V. Shulaev, D.J. Sargent, R.N. Crowhurst, T.C. Mockler, O. Folkerts, A.L. Delcher, P. Jaiswal, K. Mockaitis, A. Liston, S.P. Mane, et al. The genome of woodland strawberry (*Fragaria vesca*). *Nature genetics*, 43(2):109–116, 2010.

[46] 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*, 2(12):e1326, 2007.

[47] R. Velasco, A. Zharkikh, J. Affourtit, A. Dhingra, A. Cestaro, A. Kalyanaraman, P. Fontana, S.K. Bhatnagar, M. Troggio, D. Pruss, et al. The genome of the domesticated apple (*Malus x domestica* Borkh.). *Nature genetics*, 42(10):833–839, 2010.

[48] 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*, 15(8):1127–1135, 2005.

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

[50] Andrey Zharkikh, Michela Troggio, Dmitry Pruss, Alessandro Cestaro, Glenn Eldrdge, Massimo Pindo, Jeff T. Mitchell, Silvia Vezzulli, Satish Bhatnagar, Paolo Fontana, Roberto Viola, Alexander Gutin, Francesco Salamini, Mark Skolnick, and Riccardo Velasco. Sequencing and assembly of highly heterozygous genome of *Vitis vinifera* L. cv Pinot Noir: Problems and solutions. *Journal of Biotechnology*, 136(1-2):38 – 43, 2008. ISSN 0168-1656. doi: 10.1016/j.jbiotec.2008.04.013. URL `http://www.sciencedirect.com/science/article/pii/S0168165608001880`. Genome Research in the Light of Ultrafast Sequencing Technologies.

[51] Michael C Zody, Zhaoshi Jiang, Hon-Chung Fung, Francesca Antonacci, LaDeana W Hillier, Maria Francesca Cardone, Tina A Graves, Jeffrey M Kidd, Ze Cheng, Amr Abouelleil, et al. Evolutionary toggling of the MAPT 17q21. 31 inversion region. *Nature genetics*, 40(9): 1076–1083, 2008.