UNIVERSITY OF CALIFORNIA, SAN DIEGO

Combinatorial Methods in Computational Genomics: Mammalian Phylogenetics Using Microinversions and Fragment Assembly With Short Reads

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in
Bioinformatics

by

Mark Chaisson

Committee in charge:

Professor Pavel A. Pevzner, Chair
Professor Vineet Bafna
Professor Joesph Ecker
Professor Terry Gaasterland
Professor Xiaohua Huang
Professor Glenn Tesler

2008

The dissertation of Mark Chaisson is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2008

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

The work in this thesis encompasses two fairly disparate topics in Bioinformatics: discovery of characters for phylogenetics, and DNA fragment assembly. Although these two topics are effectively never mentioned together in Bioinformatics, they have an intrinsic and slightly transparent link in my studies: they both involve comparisons of sequences of length on the scale of the size of genomes.

In our phylogenetics analysis we consider sequences that are inverted in the genomes of two different species as ideal characters for phylogenetic reconstruction. To detect such characters we perform pairwise alignment genomes (or large Co-linear subsequences of genomes), and allow for inversion rearrangements.

The motivation for this work came after we performed the first study on short read fragment assembly and as an outcome of developing methods to perform comparison of assemblies to judge assembly quality. When assemblies are compared, they often agree in sequence composition of non-repetitive areas of the genome, however often the order and orientation of certain segments often differ between assemblies, and are considered 'misassemblies'. To detect misassemblies, it is necessary to perform a genome-scale pairwise alignment of two sequences that allows for inversion and transposition of sequences.

Our studies on short read fragment assembly began after the initial publications of protocols such as Pyrosequencing and Polonies, but before any real data was produced, allowing freedom to investigate other topics. We found that our tools for comparing genomes could discover inversions between human and chimpanzee, which was just beginning to be released in draft form at the time. The same tool could be used to detect inversions between human and mouse, and so it seemed possible to detect the orientation of very short sequences across long evolutionary distances. Although many more developments were required in the end to extend this to detecting inversions in distantly related genomes, this resolves the link between the two fairly disparate topics of this thesis.

Our method for phylogenetic reconstruction using inversions as characters was published in the Proceedings of the National Academy of Sciences around the time that other publications regarding use of characters that are *rare*: events such as inser-

tions/deletions and inversions that are so infrequent that they are not likely to recur nor revert, and thus do not suffer from homoplasy. Rare characters were not used in the past because they require massive amounts of sequence to detect enough characters for phylogenetic reconstruction, and without initiatives directed by the NHGRI and later developments by high-throughput sequencing, would never had been possible. Since sequence data is growing at an exponential rate, rare characters will grow in importance in phylogenetic reconstruction.

In 2007 we were approved on a grant with Dr. Xiaohua Huang to develop a method for short read fragment assembly, based on previous work Dr. Pevzner had done on fragment assembly, and our initial study of short read fragment assembly. This presented us with two options: continue development on methods for phylogenetic reconstruction as well as short read fragment assembly, or focus solely on fragment assembly. We began attempting to work on both, however it became clear that there was going to be fierce competition in short read fragment assembly, and diversion of attention from this topic would be a competitive disadvantage.

Thus, the work of this thesis is divided into two parts. Part I is on our development of inversions as characters for short read fragment assembly, the results for building phylogeny, and finally a study we did that redefined the genome-scale map of inversions between human and chimpanzee. Part II is on our development of the short read fragment assembly tool: EULER-SR.

ACKNOWLEDGEMENTS

First and foremost I would like to thank Dr. Pevzner for his guidance and mentorship during my time as a graduate student at UCSD.

I also thank the other faculty who have participated in projects or contributed many helpful discussions. Dr. Haixu Tang for introducing me to the field of fragment assembly, and for providing constant advice during my work on my thesis. I thank Dr. Glenn Tesler for his frequent and perspecatious discussions on modifications of fragment assembly for short reads.

I am appreciative of my fellow classmates in Bioinformatics for our mutual support and camraderie for their work in trying to define and make a success this emerging field. My labmates, Shaojie Zhang, Jocelyne Bruand, Qian Peng, and Fjola Bjornsdottir for their great company and support. I thank past members of the lab: Neil Jones, Sam Payne, Degui Zhi, and Nuno Bandiera for invaluable scientific advice.

I thank my family for their love and support, and I thank my friends for being family. They were always there to remind me what lies in life outside the lab.

Chapter 2, in full, is a reprint of an article published in 2006 (Chaisson MJ, Raphael BJ, Pevzner PA, *Proceedings of the National Academy of Sciences*, 2006, 103(52):19824-9).

Chapter 4, in full, is taken from an article published in 2004 (Chaisson MJ, Tang HA, Pevzner PA, *Bioinformatics*, 2004, 20(13):2067-74).

Chapter 5, in full, is taken from an article published in 2008 (Chaisson MJ, Pevzner PA, *Genome Research*, 2008, 18:324-330).

Chapter 6, in full, is taken from an article in submission (Chaisson MJ, Brinza D, Pevzner PA, 2008).

VITA

2001	Bachelor of Science University of California, San Diego
2002–2008	Research Assistant, University of California, San Diego
2008	Doctor of Philosophy University of California, San Diego

PUBLICATIONS

Mark J Chaisson, Haixu Tang, and Pavel A. Pevzner "Fragment assembly with short reads". Bioinformatics 20(13). (2004) pp 2067-74.

Mark J Chaisson, Benjamin J Raphael, and Pavel A. Pevzner "Microinversions in mammalian evolution". PNAS 103(52) (2006) pp 19824-9. Mark J Chaisson and Pavel A Pevzner "Short Read Fragment Assembly of Bacterial Genomes". Genome Research. 18:324-330, 2008.

FIELDS OF STUDY

Major Field: Bioinformatics
Studies in Bioinformatics.
Professor Pavel A Pevzner

ABSTRACT OF THE DISSERTATION

Combinatorial Methods in Computational Genomics: Mammalian Phylogenetics Using Microinversions and Fragment Assembly With Short Reads by

Mark Chaisson
Doctor of Philosophy in Bioinformatics
University of California, San Diego, 2008
Professor Pavel A. Pevzner, Chair

In this dissertation, two problems are considered: phylogenetic reconstruction using micro-inversions as phylogenetic characters, and *de novo* fragment assembly of short reads. It is shown in Part 1 that one may use inversions at orthologous loci detected in pairwise alignments to reconstruct phylogeny. In Part 2, a method to assemble "short", 30-100 base reads is presented. First, a study is shown that was the first to show that it is possible to perform *de novo* assembly of short (simulated) reads. Next, a method, EULER-SR is presented to assemble short reads produced by the 454 Life Sciences platform. Finally, the method EULER-USR is presented to perform mate-paired assembly of short, 35-50 base reads, and to perform error correction on reads with erroneous suffixes to recover a longer usable portion of an otherwise short read.

1

Part I Introduction: Phylogenetics

Nothing in Biology Makes Sense Except in the Light of Evolution
Theodosius Dobzhansky, 1973

This often cited quotation is given in the context of explaining why a subpopulation differs from others, how a selective advantage arose, or seemingly any circumstance involving an inference based on a comparison of a group of species that have evolved from the same common ancestor. To make any inferences, however, the evolutionary relationships that are used should be correct. The catch is, except for a few isolated examples where evolution may be measured in a laboratory such as the evolution of antibiotic resistant strains of bacteria, it is impossible to actually observe. This begs the question: How may one first shed light on evolution? This question is addressed by the field of Phylogenetics with the approach that given a set of measurements on a group of extant species, and a model of how we believe evolution to occur, one may build a phylogeny that most likely encodes what happened over the course of evolution.

The group on which a phylogeny is reconstructed is a *taxa*, and the features that are measured *characters*. Until the work of Emile Zuckerkandal and Linus Paling, phylogenetic reconstruction was performed using morphological characters. Morphology may involve discrete characters, such as number of toes, however there is typically there is much uncertainty in this as morphology is the result of complex developmental pathways. Considerable debate may arise when inferences are made using characters with

uncertainty. The efforts to discover the evolutionary relationship between whale, hippopotami, and pigs are a classical example of this. Careful morphological measurements such as dental patterns and also blatant similarities such hind legs supported grouping hippopotami Suiforms (including pig). The resolution of the controversy began in the early 1990's when several groups used protein sequences from hippopotami, and several species from Suiforms and cetecae (whales) to derive a molecular phylogeny that placed hippos and cetacae as sister groups. It is argued that sequences evolve differently from morphology, (some) without natural selection, and provide for a more objective model of evolution.

Sequence based phylogenetics began in 1962 when Pauling and Zuckerkandl produced a primate phylogeny with hemoglobin sequences [154]. The motivation was that morphology is a complicated result of different protein functions, so sequence based phylogenies should be more accurate. Much progress has been made in the field of sequence-based phylogenetics, from parsimony to probabilistic methods. An example of the sophistication of sequence-based phylogenetics is a phylogeny of 44 mammals using Bayesian phylogenetics [88]. Sequence based phylogenetic reconstruction is subject to the constraint that it best models the underlying process of nucleotide or amino acid sequence evolution. Unfortunately, the underlying mechanism for nucleotide evolution may vary. Many Muridae sequences are more divergent from primate sequences due to the lower accuracy of Muridae DNA polymerase than Laurasiatherian, yet Muridae are thought to be more closely related to primates. Furthermore, the small character space (the set of amino acids or nucleotides) suggests that homoplasy, independent evolution to the same state, is more likely. Phylogenetic reconstruction may be more accurate using characters less subject to homoplasy and for which the underlying mechanism for change is constant.

Phylogenetic methods may be largely classified according to three approaches: distance, likelihood, and maximum parsimony. All methods consider a set of n species for which one would like to derive phylogeny using m characters, so each species $s_1, \ldots, s_n$ is encoded by an m element character vector. The phylogeny is encoded as a tree T where the leaves of the tree are extant species, and the parent of any node is the immediate ancestor. Distance based methods define a distance measure on pairs of

species $d_{i,j} = Distance(s_i, s_j)$, and form trees by joining species of least distance. A popular method is the Neighbor-Joining method [121], where pairs of leaf vertices are joined, and replaced by a single ancestral species representing a combination of the two until one vertex (the root of the phylogeny) remains. Methods using likelihood require a model of character evolution $\mathcal{M}$ that describes the likelihood of one character evolving to another. The likelihood methods find a tree T that maximizes $L(T|S, \mathcal{M})$. Finally, maximum parsimony methods seek to find a similar edge-labeled tree where the total number of transitions in the tree are minimized.

The character matrices used in molecular phylogenetics is a multiple alignment of an orthologous sequence. Each column in the multiple alignment is a character. This indicates that the type of alignment that is used affects the outcome of the reconstruction. An example in [149] demonstrated that a certain protein sequence aligned with seven different alignment methods produced six different trees using likelihood methods.

1.A Rare characters

A fairly radical alternative to the standard molecular phylogenetics of the last 40 years has emerged with *rare characters*. The motivation is simple: rather than using somewhat uncertain point mutations as characters, find events that in all likelihood are irreversible but also easy to define. Events that have been considered for rare rare characters are gene duplications [32], retropositions [95, 8], and micro/macro genome rearrangements [15]. The drawback to using rare characters is implied in the name: it may be necessary to search large amounts of sequence in order to find relatively few characters for phylogenetic reconstruction.

Some of the first investigations performed using rare characters used Short Interspersed Nuclear Elements (SINEs) retropositions to build phylogeny. SINEs are short, 100-500 base sequences that are copied from the genome and re-inserted. They are good candidates for rare characters because they are believed to not be excised from the genome, and are unlikely to insert at the exact orthologous position independently in different lineages. The usefulness of SINE elements as phylogenetic markers was first proposed by Norihiro Okada in 1991 [96], and practically demonstrated by resolving

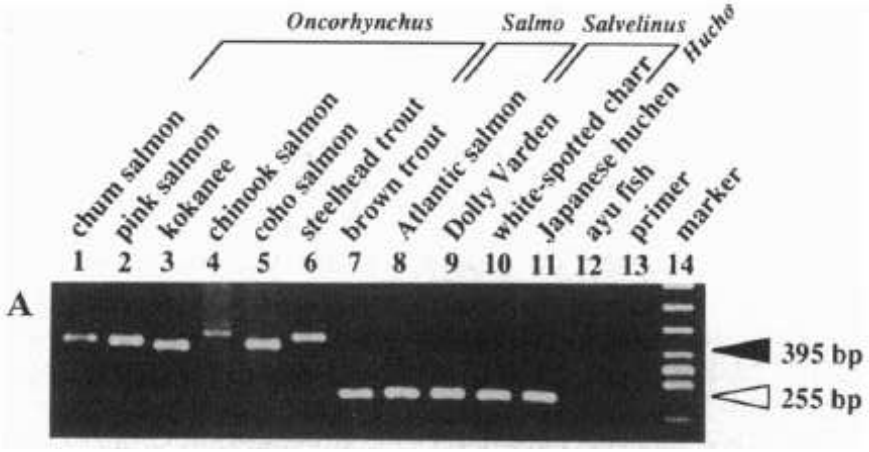


Figure 1.1 Manual validation of a transposition in a clade of Salmon [87]. This may be used as a certificate to separate the *Onchorynchus* lineage from *Salmo* and *Salvelinus* lineages.

closely related species of Salmonidae [87]. The presence/absence of SINE elements was meticulously detected using PCR and gel electrophoresis. An example of a confirmed SINE insertion is shown in Figure 1.1. After measuring presence/absence of a number of SINE elements, a phylogeny was constructed (by hand) using the assumption that elements are either present,absent, or unconfirmed (due to deletion). Once gained a SINE could only become unconfirmed. This allows one to mark edges on the phylogeny where certain SINE insertion events occurred, as shown in 1.2.

This method has been used by Okada to resolve small disputed phylogenies such as the whale/hippotamus/pig trichotomy [95], and by others for primate phylogenies [134, 9]. The drawbacks to the Okada method are that it only validates phylogeny of closely related species because of the use of PCR to validate insertion events. Furthermore, the assumption that mobile elements never integrate at the same location, up to the resolution of the method for detecting them may not be true when the mobile element family has been very active and there is sequence specific insertion bias [55].

When methods for phylogenetic reconstruction using SINEs were formulated one and a half decades ago they were formulated using a fairly small number of characters because there was relatively little sequence information available in species other

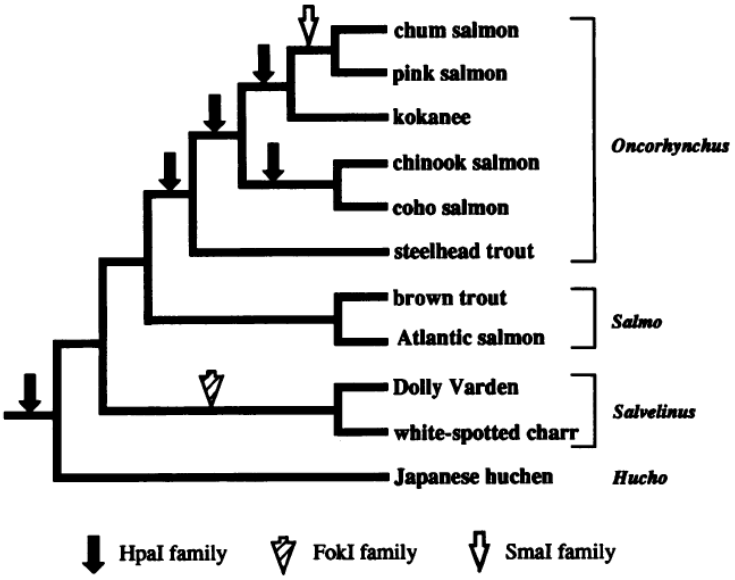


Figure 1.2 The phylogeny of salmon constructed using SINES by Okada [87]. This is one of the first phylogenies built from rare molecular characters.

than a handful of model organisms. Because of the development of large scale sequencing facilities such as the Washington University Sequencing Center, and the support of the National Human Genome Research Institute (NHGRI) and the NIH Intramural Sequencing Center (NISC) to gain functional understanding through comparative genomes, a large number of targeted orthologous regions have been sequenced from a diverse set of mammals [143].

After annotating known repeats using RepeatMasker with Repbase [65], one may detect orthologous repeats using pairwise alignments. Methods to align pairs of long (over 1-megabase) mammalian sequences were perfected to compare the human and mouse sequences [130], and used to build a multiple alignments over collections of long sequences [129]. This allows one to trace the presence/absence of a transposable element across many species, and to thus create large datasets using repeat insertions as phylogenetic markers.

A large scale study using orthologous transposable elements was performed by Ali Bashir and others in 2003 [8]. As prior methods for phylogenetic reconstruction were largely *ad-hoc*, manual methods based on the assumption that the characters did

not exhibit any homoplasy, no computational methods had been defined to construct phylogeny using computationally derived, large-scale datasets. This study had several important results. First, they showed that insertion characters do exhibit a small degree of homoplasy. Fortunately, due to the rare frequency of homoplasy, the authors were able to develop robust methods to detect and remove homeoplasic characters prior to reconstructing phylogeny. Finally, the authors developed a robust method for reconstructing phylogeny using insertions as directed markers, when the presence/absence of some markers are not known for all species: presence/absence/*unknown* were the three possible states for a character.

The main drawback to this method is that it is very difficult to reliably detect transposable elements across large evolutionary distance.

1.B Genome rearrangements

Genomes evolve at scales that range from point mutations to rearrangements of large portions of chromosomes. Early observations of the latter were found in 1988, when it was observed that the gene content for mitochondrial genomes of cabbage and turnip are largely identical except in shuffled order [97]. When the gene content is identical, one may transform the gene order of one species into the order of another using reversals. Using parsimony, one may reconstruct what was likely the series of reversals that took place during evolution by finding the minimal number of rearrangements to transform one gene order into another. A wide body of research exists for finding optimal rearrangement scenarios between two genomes [125, 50].

It was proposed by David Sankoff that rather than reconstructing the rearrangement history between two species, if the most parsimonious rearrangement of a group of species were reconstructed, one may use that to build phylogeny [126]. Unfortunately, this involves solving an NP-hard problem [22]. Several heuristics have been proposed to do this [15, 85].

The main problem with this approach is that since it involves solving an NP-Complete problem, different heuristics may produce different solutions. Furthermore, most methods are based on parsimony, rather than likelihood of rearrangements. An

analysis performed by Larget et al [72] demonstrated an instance where a method to estimate the most likely rearrangement scenario also found a scenario that had fewer rearrangements than the maximum parsimony method MGR.

The difficulty with reconstructing rearrangement scenarios is that rearrangement events are overlapping. When the rearrangement events do not overlap (micro-inversions), it is shown in the following chapter that one may reconstruct gene order in linear time.

1.C Micro inversions

In this work we propose that micro-inversions may be added to the growing base of rare characters for phylogenetic reconstruction. Unfortunately, this is hardly a novel idea; in 1936 it was used by Dobzhansky and Sturtevant to reconstruct the phylogeny of drosophila subspecies with inversions discovered from banding patterns in stained chromosomes. The banding patterns of the polytene chromosomes in drosophila may be viewed under a microscope (reproduced in Figure 1.3,left), which allowed Dobzhansky and Sturtevant to characterize genome structure well before biochemical techniques to do so were invented. The micro-inversions were used as the first data for molecular phylogenetics to reconstruct the phylogeny of drosophila subspecies in the western United States [34] (Figure 1.3, right).

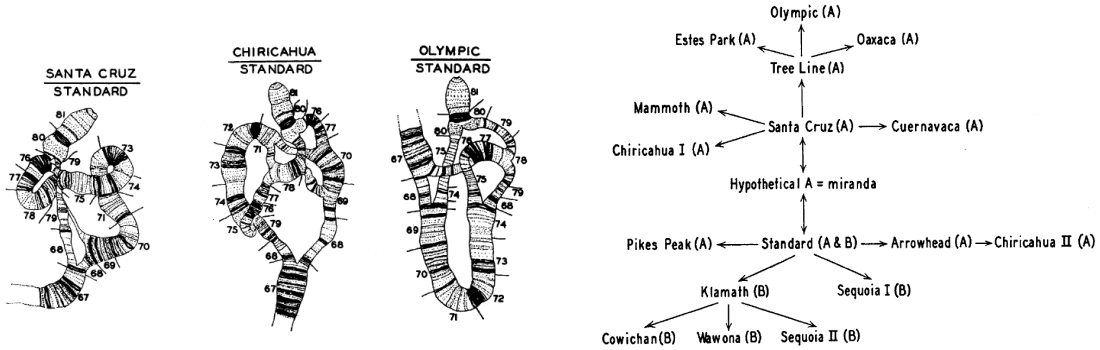


Figure 1.3 (*(left)* The banding patterns used to build a Drosophila phylogeny. *(right)* A phylogeny of Drosophila subspecies created using inversions found in cytogenic banding patterns. Figure reproduced from [34]. Figures reproduced from [34].

1.D Mammalian phylogeny

In the following chapter we use micro-inversions to reconstruct the phylogeny of a diverse group of mammals. The importance of having an accurately reconstructed mammalian phylogeny is underscored by the inferences made using comparative genomics [30], as well as having a long standing importance to basic understanding of mammalian evolution. Many attempts have been made to infer the mammalian phylogeny [115, 88], yet even with sophisticated methods, controversy over areas of the tree as small as the order of evolution of human,dog,and mouse still exist [21].

Rare characters were used in [8] for the reconstruction of the mammalian phylogeny, and in [62] for small branches. Our method for phylogenetic inference based on micro-inversions adds to the aresnal of tools for phylogenetic reconstruction.

2

Micro Inversions in Mammalian Evolution

2.A Introduction

In this chapter we propose a new approach for identifying micro-inversions across different species and show that micro-inversions provide a new source of low homoplasy evolutionary characters. These characters may be used as "certificates" to verify different branches in a phylogenetic tree turning the challenging problem of phylogeny reconstruction into a relatively simple algorithmic problem. We estimate that there exist hundreds of thousands of micro-inversions in genomes of mammals from comparative sequence projects, an untapped source of new phylogenetic characters.

Chromosomal inversions have been used as phylogenetic characters since the time of Dobzhansky and Sturtevant (1938). Recent comparisons of whole mammalian genomes have revealed a surprisingly large number of micro-inversions [108, 68]. While this was first met with scepticism and was attributed to assembly errors and alignment artifacts, recent comparative study of human and chimpanzee genomes convincingly proved that micro-inversions are indeed wide-spread [42]. We therefore decided to perform a fine-grained search for inversions across many mammals in the greater CFTR region. This is a 1.8 Mb region on chromosome 7 in the human genome that encompasses the CFTR gene, and its many neighboring genes, that was sequenced for the ENCODE

project [29, 143]. We found that micro-inversions are frequent across all species and occur at roughly one micro-inversion per megabase per 66 million years of evolution. We show that micro-inversions have low homoplasy, and thus provide ample characters for phylogenetic studies.

Our work follows in the steps of the pioneering work by Okada's group [94] and Lake's group [117] that demonstrated the power of repeat-based and deletion-based characters to resolve difficult phylogeny problems that the traditional point mutation analysis failed to resolve. The repeat-based and deletion-based approaches, while very successful, have some drawbacks as reviewed in [55]. However, Bashir et al., 2005 [8] and Kriegs et al, 2006 [71] recently demonstrated that many repeat-based characters may be extracted from genomic sequences to alleviate these drawbacks and to resolve some existing controversies. Our work reveals a new source of low homoplasy phylogenetic characters that complement these previous studies in two respects. First, micro-inversion homoplasy (if any) may be detected by the *breakpoint graph* analysis [4] and such characters can be simply deleted from further consideration without affecting the tree reconstruction algorithm. Second, micro-inversions may be identified as long as there is a detectable sequence similarity thus not necessarily limiting the comparison to close species as in the case of repeats and deletions. Indeed, Bourque et al., 2005 [16] documented many micro-inversions between human and chicken genomes while Fischer et al., 2006 [43] found many micro-inversions between yeast genomes that are molecularly as diverse as the entire phylum of chordates.

While micro-inversions represent powerful evolutionary characters, their detection is far from being simple. A naïve approach is to detect reverse-strand local alignments between orthologous sequences. However, reverse-strand local alignments may also be caused by palindromes and inverted repeats (Figure 2.1), ubiquitous genomic features that do not reflect any variations in the genomic architecture between two genomes, i.e., they may be detected within a single genome without a need to align to another genome. Reverse strand alignments may also be detected in inverted transpositions (Figure 2.1) and more complex interleaving rearrangement events (not shown). The computational challenge of distinguishing between micro-inversions and other genomic features is not widely appreciated leading to an implicit assumption that whole-genome reverse-strand

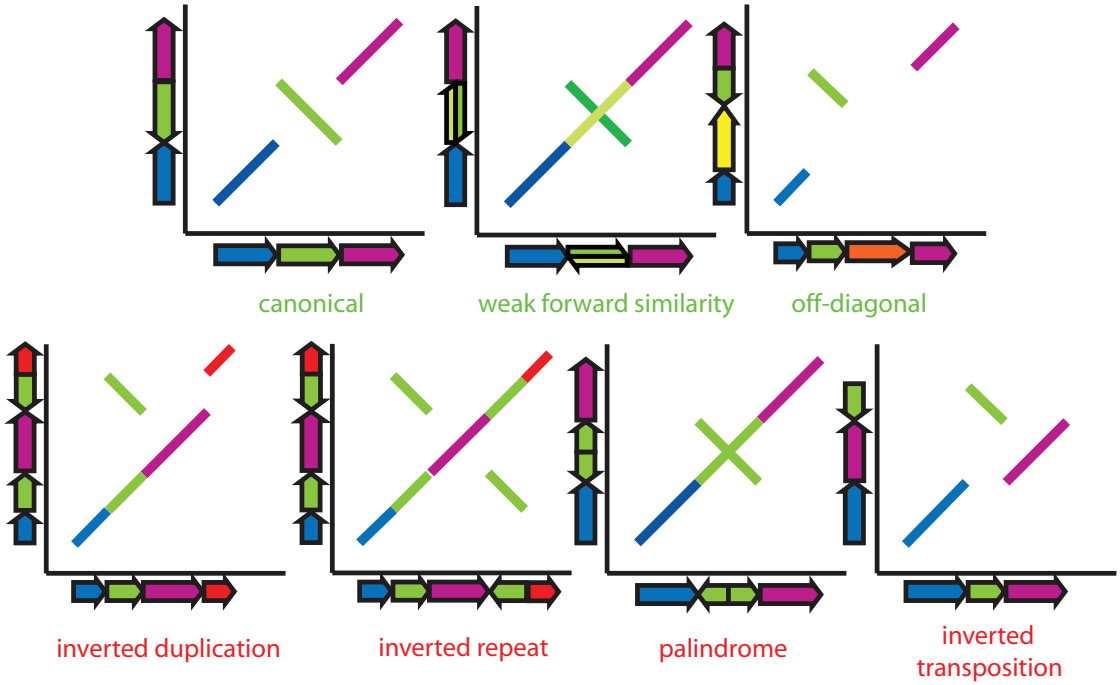


Figure 2.1 Diagrammatic dot-plot of inversions (top) and genomic structures that are often misclassified as inversions (bottom). The top row shows alignments that are retained as inversions: canonical inversions, inversions with spurious forward similarities, and off-diagonal inversions that have been shifted by insertions/deletions or have highly diverged segments that elude similarity search. The bottom row represents inverted duplications, inverted repeats, palindromes, and inverted transpositions.

alignments retained in a net by the UCSC chaining and netting algorithms [68] provide a universal solution to the characterization of micro-rearrangements. Chaining and netting combine optimal ordered sequences of pairwise alignments to create a genome-scale alignment that allows for gaps and inversions (see Appendix A for additional details). We developed a method, *InvChecker*, to find inversions in the CFTR region, and applied it to the human and chimpanzee genomes to show that approximately 80% of the 1576 putative micro-inversions recently found in [42] are repeat-induced artifacts. At the same time we uncover 167 human-chimpanzee micro-inversions missed in [42]. [1]

These findings reveal some limitations of chaining and netting [68] as a micro-inversion detection tool in [42]. This is not a criticism of this method but rather an

[1]This is a long footnote.A nighttime stream of fast-moving air in the Great Plains separates cool, moist air near the ground from drier, warmer air above. The simulation found that the turbines would catch this nocturnal jet, with ensuing turbulence and vertical mixing. When the upper air mass reached the surface, the warming and drying effect would be significant, the model suggested.

indication that accurate validation, parameter setting, and post-processing are necessary to extract micro-inversions from netted alignments. The chaining and netting algorithms [68] were carefully designed as a compromise between providing a simple and intuitive representation of rearrangements on one hand, and reflecting all complexities of the rearrangement process on the other hand. This representation, while extremely useful, does not attempt to model complex rearrangements (e.g., overlapping inversions) in full generality. We further illustrate the use of micro-inversions evolutionary studies.

2.B Results

2.B.1 Identification of micro-inversions

We searched for micro-inversions in 15 genomes with the nearly finished greater Cystic Fibrosis Transmembrane conductance Regulator (CFTR) region[2]: human, chimpanzee, baboon, macaque, marmoset, galago, rabbit, mouse, rat, cow, dog, platypus, opossum (*Monodelphis domestica*), and hedgehog [143]. These sequences (May 2005 ENCyclopedia Of DNA Elements ENCODE release [66]) average 1.84 Mb in length. Sequences comprised of multiple contigs were ordered and oriented according to alignments with the human sequence. Each sequence was repeat masked with Repeat Masker using both the Repbase and species-specific RepeatScout libraries [65, 113].

While many putative inversions are detected using chaining and netting, simple analysis of the netted files and reciprocal best hits as in [42] has serious shortcomings. First, some more divergent inversions are missed in netted alignments because netted alignments have a tendency to favor the direct alignment even if the reverse alignment is more statistically significant, albeit miniscule. For example, two micro-inversions were found in the alignment of human and chimpanzee in the greater CFTR region that both remained undetected in whole genome nets (one of them is shown in Figure 2.2). Second, BLASTZ based alignments that are processed by chaining and netting miss some "ancient" inverted segments. For example, while a segment in human may have a detectable similarity in mouse but not in rat, aligning the mouse segment against the rat genome may lead to detecting similarity between the human and rat sequences.

[2]We limited our analysis to genomes with no more than 200,000 unfinished base pairs.

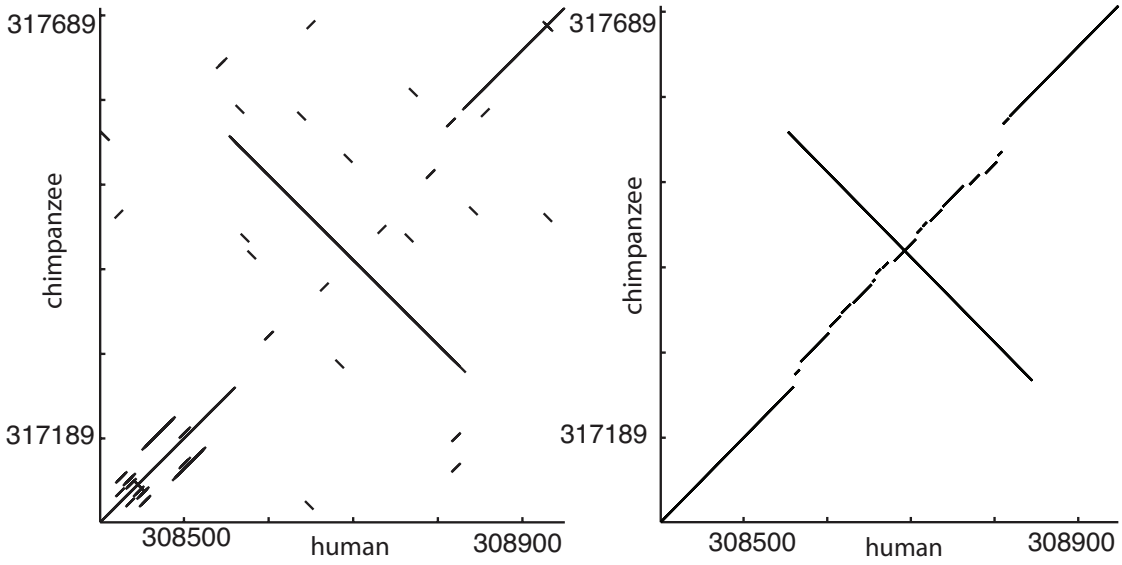


Figure 2.2 The dot plot (left), and the corresponding BLASTZ alignments (right) of a 290 bp micro-inversion that is not detected using netted alignments of human and chimpanzee (false negative). The dot-plot on the left clearly shows the presence of a micro-inversion. However, the gap spanning the forward alignment is small enough to be closed by chain-and-extend alignment although the score of the direct strand alignment is not statistically significant, particularly when compared to much higher score of the reverse strand alignment.

Third, genomes typically have many palindromes and inverted repeats that may mimic micro-inversions thus misleading the netting and chaining algorithms. An example of this is shown in Figure 2.3 that represents an inverted repeat that is misclassified as a micro-inversion.

Figure 2.1 shows seven genomic dot-plots, only the first three of which represent micro-inversions. The four others may be mistakenly classified as micro-inversions if one only considers reverse strand alignments. On the other hand, the inversion shown in the second and third dot-plots are often missed in chaining and netting. Detecting inversions is not unlike the problem of finding orientations of highly diverged synteny blocks (see [110]) that requires a careful computational analysis. To address these complications, we developed a program, *InvChecker*, that analyzes artifacts shown in Figure 2.1 by searching for inversions in all reverse strand alignments, rather than those simply retained in a net (see Appendix A).

The pairwise representation of micro-inversions detected by our analysis hides

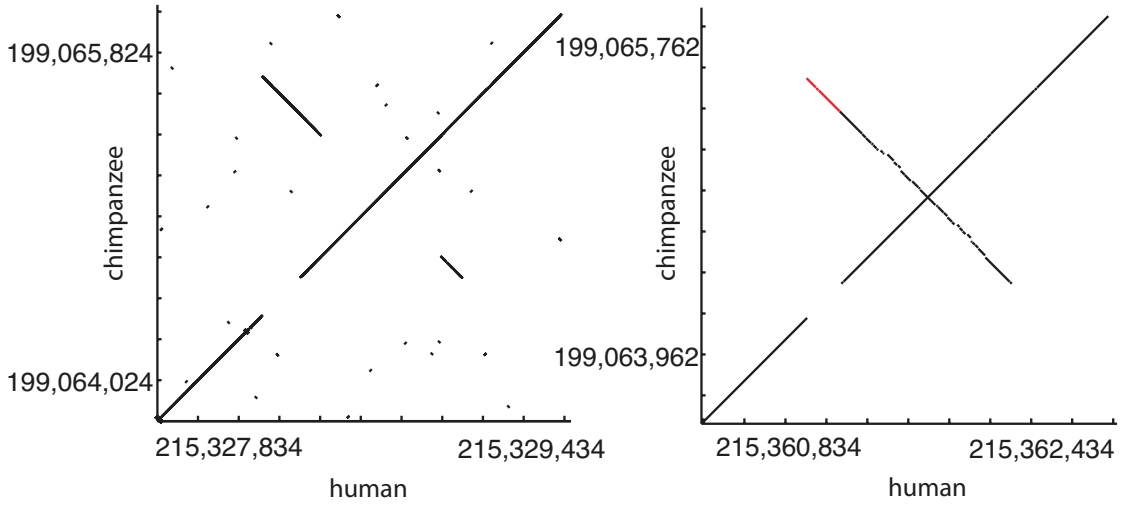


Figure 2.3 The dot plot (left), and the corresponding BLASTZ alignments (right) of an inverted repeat that is misclassified as a putative micro-inversion by chaining and netting (false positive). The red segment in the figure on the right corresponds to an inversion taken from the list of inversions presented by [13]. The undetected forward alignment, directly below red segment is caused by a short run of "N" nucleotides in the chimpanzee assembly. The same effect may be caused by some regions in inverted repeats/palindromes being more diverged than other thus eluding alignment algorithms.

the fact that are insertions/deletions and alignment artifacts affect in each genome that make it difficult to rigorously define the term "inverted loci" (orthologous regions involved in inversions) across multiple species. The intuitive definition of inverted loci as a set of such orthologous regions across all species is somewhat imprecise since these regions may differ in length (due to deletions) and may include diverged non-alignable parts making it difficult to construct their multiple alignment [10]. Pairwise inverted regions between species i and j form a set of regions S_{ij} in species i and a set of regions S_{ji} in species j. The union of all such regions over all pairs of species is denoted $\bigcup S_{ij}$ (the union is taken over all indices i and j). This set represents the set of all regions in all genomes that were subjected to rearrangements (Figure 2.4). The exact endpoints of the regions in the sets S_{ij} and S_{ik} may vary widely (for $j \neq k$) and therefore the set $\bigcup S_{ij}$ provides a more accurate estimate of the span of the inversions than individual sets S_{ij}. We remark that while a micro-inversion between two species A and B may be easily detectable, the inverted region between species B and C may be too diverged to pass the alignment threshold. However, if one may align the corresponding regions in A and

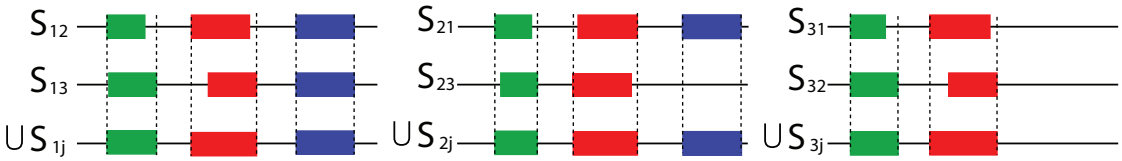


Figure 2.4 Defining the boundaries of inversion loci through the union of the boundaries of inversions found in a set of pairwise alignments.

C, the inversion between B and C may be confirmed. To address this complication, for every continuous region in $\bigcup S_{ij}$ we use a more sensitive search to find all similar regions in other species resulting in an extended set of species in which an inversion locus is detected. We iteratively use the extended set of regions to find divergent loci that were undetected in the pairwise comparisons until no new loci are discovered. This allowed finding inversions in related yet divergent species and resulted in a 40 percent increase in the number of regions found as compared to the set $\bigcup S_{ij}$.

All loci are expected to be present in each species (in direct or reverse orientation) unless (i) the locus is in a gap in an assembly, (ii) the locus was deleted in the course of evolution, or (iii) the locus is so diverged that it escapes the detection by sequence alignment. As a result, we find only 520 out of the $68 \cdot 15 = 1020$ possible regions in all inversion loci in 15 mammals.

We also detect a small number of regions that show evidence of overlapping micro-inversions like: $A\overline{BCD}E \rightarrow$ A-$\overline{\text{D-C}}$-BE $\rightarrow$ ACD-BE in the human-baboon comparison. Although such micro-rearrangements are filtered out by *InvChecker* and are not considered as characters for phylogenetic reconstruction, they are perfectly suitable for phylogenetic analysis (work in progress).

2.B.2 Removing conflicting micro-inversions

Ideally, each inversion loci yields a valid evolutionary character. However, in rare cases spurious alignments and overlapping inversions may produce ambiguous characters that need to be detected and removed before the tree reconstruction begins. We may remove ambiguous micro-inversions prior to tree reconstruction using two methods: an *alignment consistency test* that is based on the consistency of alignments within a single inversion loci, and the *four gamete test* that is based on consistency of pairs of

inversion loci.

The alignment consistency test checks that the parity of a putative inversion is consistent across multiple species. For example, if a segment in species A is inverted relative to species B, and the same segment in B is inverted relative to a segment in C, then the segments in A and C should have the same orientation. To check alignment consistency within a given inversion locus, we construct an *inversion graph* for each inversion locus: vertices correspond to the inversion locus in each species; red edges connect vertices whose loci are in opposite orientation; and blue edges connect vertices whose loci are in the same orientation (Figure 2.5). We determine the relative orientation of two inversion loci by comparing the local alignment scores of the inversion loci in the forward and reverse orientations to the (empirically determined) expected alignment score of two random sequences of the same length. The orientation is determined by the alignment that has a score k times greater than the random alignment scores, where k is a parameter. Lower values of k allow one to analyze more divergent sequences, but are more prone to errors. Furthermore, sequences that align with scores above k in both orientation (as is the case with ambiguous loci boundaries such as partially deleted loci) are not assigned an edge. When all alignments for an inversion locus are consistent then all cycles in its inversion graph should have an even number of red edges (in particular, red edges form a *bipartite graph*). Inversion loci that violate the "even number of red edges in a cycle" condition are discarded. We found that for $k = 2$ there are no cycles violating the "even number of red edges" condition.

The inversion graphs that do not violate the "even number of red edges in a cycle" condition (e.g., all inversion graphs for $k = 2$) may be used to derive evolutionary characters. The vertices of such graphs may be partitioned into two disjoint sets such that every path between vertices from two sets has an odd number of red edges (loci in one set are inverted as compared to loci in another set). We arbitrarily assign "direct" orientation to all loci in the first set and reverse orientation to all loci in the second set. Orientation is encoded in character vectors by assigning an orientation 1 to species on one side of the graph, and 0 on the other. We also assign ? to species outside the connected component (Figure 2.5, right). Combining all inversion loci results in an $n \times m$ matrix C (with 0s, 1s, and "?"s) for m character vectors and n species, shown in Figure 2.6,

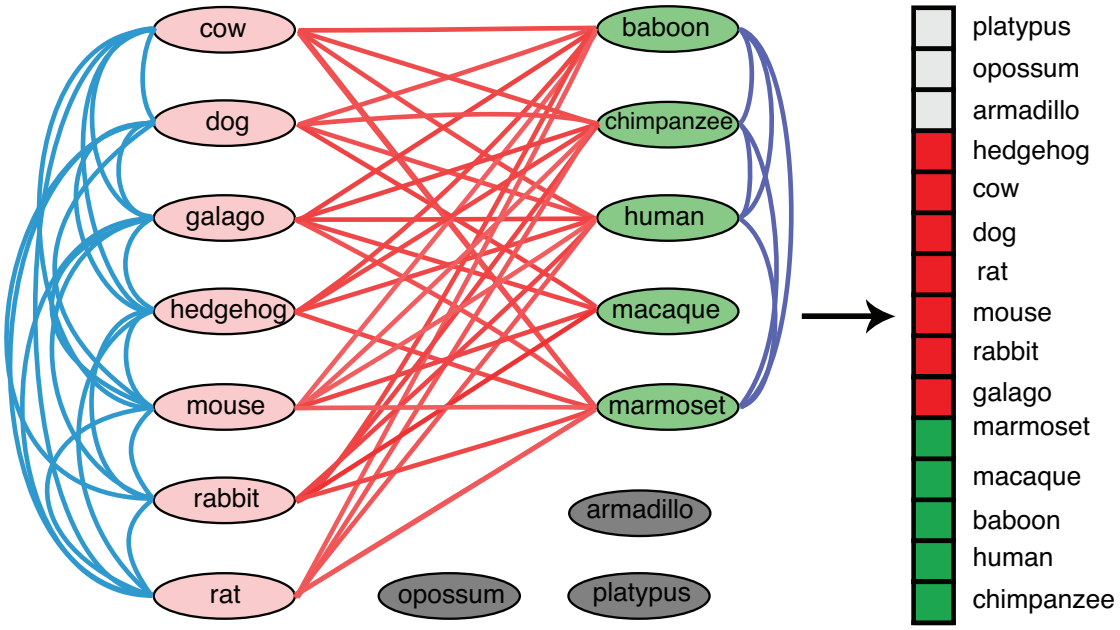


Figure 2.5 The inversion graph (*left*) and corresponding character vector (*right*) for an inversion of length 1300 bp created with edges assigned as the higher scoring alignments ($k = 2$). The graph is bipartite, indicating that there are no spuriously assigned orientations. Note that the graph is not complete; in particular the inversion locus in macaque is partially deleted. It is counterintuitive that both macaque and human loci are aligned to mouse locus but not aligned to each other. It is explained by independent deletions in different regions in macaque and human that left no "common" sequence between macaque and human sequences in this locus. However, both partial sequences in macaque and human genomes may be aligned to the mouse locus.

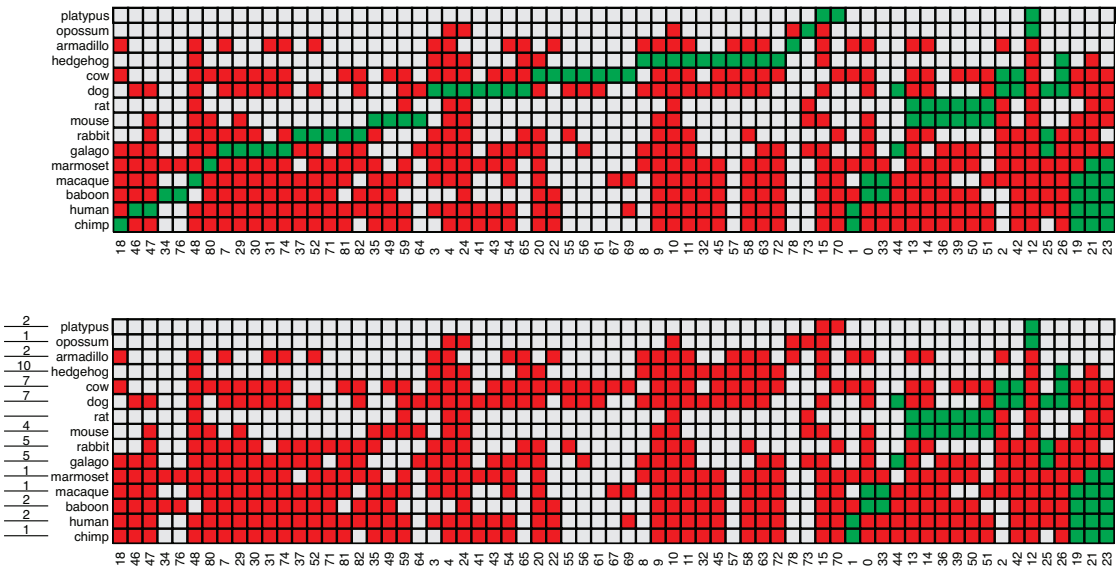


Figure 2.6 The character matrix for 67 micro-inversions in 15 species (top), and the matrix after the performing the first 49 good inversions (bottom). Each column represents an orthologous inversion locus. Red and green cells represent inversion loci in opposite orientation, and gray cells correspond to ? signs (unknown orientation). Columns with a single green cell are inversions unique to a species. The number of inversions performed on each species is shown to the left of the bottom figure.

top. The condition $C_{ij} =?$ implies that the inversion locus j is unresolved in species i. Note that the partition of each column into 0s and 1s is arbitrary and may be switched (i.e. the characters are *undirected*). The increase in the number of unresolved inversion loci with evolutionary time is attributed to both difficulties in validating such inversions, incomplete coverage of CFTR regions for some species, and the stringent parameters we use in this study (see Discussion section).

Next, we apply the four gamete test to pairs of inversion loci. We assume that the set of micro-inversions is homoplasy-free. Therefore, the micro-inversions form a *perfect phylogeny* and all pairs of characters must satisfy the *compatibility* or *four-gamete test* [58, 40]: $n \times 2$ sub-matrix of C formed by a pair of columns has the rows 00, 01, 10, and 11. Four-gamete violations may arise either from spurious alignments or from inversions that are not homoplasy-free. While in general case violating characters may be detected and removed using the *Maximal Conflict Removal* technique from [8], our original dataset of 68 inversions contained only one violation. Manual inspection of this violation revealed that it is caused by a misclassification of a rather diverged

inverted duplication (fourth diagram in Figure 2.1). This misclassified micro-inversion was removed resulting in 68-1=67 characters. Appendix B describes the distribution of these micro-inversions along the human genome.

While micro-inversions rarely re-use breakpoints, there is difference between large-scale rearrangements and micro-inversions when it comes to breakpoint re-use. Sequenced genomes revealed large breakpoint re-use imposed by different (large-scale) rearrangements that happen at the same rearrangement hot-spots [109, 89]. However, when one claims that two genome-scale rearrangements use the same breakpoint it does not mean that the breaks occur exactly the same nucleotide but rather that their exact breakpoint positions are indistinguishable at the genomic scale. The situation is very different for micro-inversions where even closely located breakpoints may be distinguished due to the smaller scale of the aligned regions. Since we found very limited micro-inversion breakpoint re-use with this higher level of resolution we postulate that repeated micro-inversions with exactly the same pairs of breakpoints (that would paraphyletically replicate micro-reversions and create homoplasy) are unlikely. The micro-inversions that share only one of their breakpoints do not represent a problem since they may be detected (via breakpoint graph [4] analysis) and discarded from further consideration.

2.B.3 Reconstructing phylogenetic trees

We first consider the case when all inversion loci are fully resolved (i.e, matrix C has no ? signs). Let $\pi_1, \ldots \pi_n$ be n be n *signed genomes* [3] that evolved by some (possibly overlapping) unknown inversions according to an unknown evolutionary tree. Without loss of generality, we assume that there was at least one inversion on every branch of the tree as zero length edges may be contracted. Every inversion creates up to two breakpoints - pairs of adjacent orthologous sequences that are out of order.

One may classify an inversion as *independent* if it creates exactly two new breakpoints (i.e., increases the number of breakpoints by 2) and does not re-use breakpoints [110]. We call the rearrangement process *independent* if all its inversions are independent, up to the resolution of the boundaries of each inversion.

If all inversions were resolved (no ? signs in matrix C) and did not re-use break-

[3] See Pevzner, 2000 [101] for a background on genome rearrangements.

points, the following variation of the perfect phylogeny theorem [19] would immediately resolve the problem of reconstructing inversion-based phylogenetic tree.

Theorem. If n genomes of length m are produced by independent inversions then both the correct evolutionary tree (up to the zero-length edges) for these genomes and the ancestral architectures of all its branching vertices may be reconstructed in polynomial time.

For the sake of completeness, we give the outline of the proof below. Let $\pi_1, \ldots \pi_n$ be n genomes that evolved according to an (unknown) evolutionary tree T and let $b(\pi_i, \pi_j)$ denote the number of breakpoints between genomes π_i and π_j. Since every rearrangement creates exactly two breakpoints, the tree path between leaves π_i and π_j accounts for $b(\pi_i, \pi_j)/2$ rearrangements. The inversion distance between these genomes is at least $b(\pi_i, \pi_j)/2$ implying that the tree T is additive [101]. Since the (unknown) tree T is additive and since the distances between its leaves are known, Zaretskii's theorem [150] implies that it may be uniquely reconstructed in linear time. An observation that the *median* [15] of every three genomes π_i, π_j, and π_t evolved by independent rearrangements is unique and may be reconstructed in linear time implies that the permutations corresponding to all branching vertices in the tree T may be uniquely reconstructed.

The above theorem does not impose any restrictions on the reconstructed tree (like parsimony) and only assumes the evolutionary process consists of independent events. This is a reasonable assumption since micro-inversions rarely re-use breakpoints.

It is straightforward to show by following the arguments used in the proof of the above theorem, that in case of independent evolution the MGR (Multiple Genome Rearrangements) algorithm [15] reconstructs the correct evolutionary tree. MGR constructs an evolutionary tree while seeking to minimize the number of inversions. However, MGR assumes that all inversions are resolved. Since micro-inversions are often unresolved for distant species we developed an MGR-like heuristic that is directed toward data with unresolved characters (the "?" signs in matrix C).

Note that a removal of an edge from a phylogenetic tree partitions the tree into two subtrees. Our goal is to reconstruct a tree and assign every character to an edge in the tree in such a way that if this edge is removed then all 1s are in one subtree, while

all 0s are in another subtree (see [137]).

Intuitively, our algorithm attempts to move "back in time" undoing micro-inversions, i.e., performing inversions of inverted loci that bring the existing species closer to the ancestral mammalian genome. This is achieved by evaluating all possible inversions for each genome, and identifying *good* inversions that bring a genome closer to the ancestral genome. Of course, the ancestral genome is unknown and therefore it is unclear how to find good inversions. However, Bourque and Pevzner, 2002 [15] argued that an inversion which brings a particular genome closer to *all* other genomes is likely to be a good inversion. If this is correct, then we do not need the ancestral genome to find good inversions. We then carry on performing good inversions in all genomes and iterate until some of genomes (e.g., A and B) do not have any loci in different orientations (converge to their most common ancestor). After A and B become identical there are no longer good inversions in A and B (since any inversion in A will make it more distant from B) and we merge A and B into a single genome (thus enabling good inversions at the next iteration) and iterate. Of course, this approach works well only for "nearly perfect" characters, and we argue that it is the case for micro-inversions.

Therefore, our MGR-like algorithm is very simple: look for good inversions in all genomes and perform them (if there are any) until some of the genomes become identical, merge identical genomes and iterate. For example, in Figure 2.6 (top) is one good inversion in chimpanzee (corresponding to the green cell in the first column), two good inversions in human (green cells in the second and third columns), two in baboon, one in macaque, etc. We "reverse" all 49 good inversions (Figure 2.6, bottom) so that some species become identical. For example, human and chimpanzee, macaque and baboon, mouse and rat, etc. become identical in Figure 2.6(bottom). The difficulty, however, is that, since some inversions are unresolved, there is a danger that some inversions may appear to be good while in fact they are not depending on the value (0 or 1) assigned to one of the "?" signs. Another danger is that some genomes may appear identical (after performing some good inversions) while in fact they are not if the ? signs are replaced by 0 or 1. Armadillo/hedgehog and platypus/opossum represent an extreme case of such potentially incorrect merges since they have a single shared inversion locus. We address the uncertainty caused by ? signs with a greedy heuristic: we postpone

merging species in any iteration if they have less than p percent resolved characters in common (where t is a threshold). For $p = 90\%$, the merging of platypus/opossum and armadillo/hedgehog will be postponed despite the fact that they represent "valid" merges. The number of remaining characters decreases as species are merged, and so merges that are postponed in an early stage are performed later.

Since our character matrices include "?" characters it is possible that there are species that are pairwise, but not transitively equivalent. Consider a simple example of species A, B, and C, with three characters in the matrix:

$$\begin{array}{cccc} A & 1 & 1 & ? \\ B & 1 & ? & 0 \\ C & ? & 0 & 0 \end{array}$$

. In this example $A \sim B$, and $B \sim C$, but $A \not\sim C$. Sequences that are highly divergent, or that contain many gaps may be missing characters that create such inconsistencies. To avoid artifacts caused by unresolved characters we merge the largest set of transitively equivalent species for which there are no inconsistencies. While this greedy heuristic is important in cases when there are a limited number of micro-inversions, it may not be necessary when more sequences are available.

After the first round of good inversions, our greedy heuristic merges human and chimp, macaque and baboon, galago and rabbit, mouse and rat, and cow and dog. Afterwards we are left with 18 characters that represent "earlier" micro-inversions (Figure 2.7a). Again, there exists 1 good inversion in the human+chimpanzee ancestor (1st row), 2 good inversions in the macaque+baboon ancestor (2nd and 3rd rows), 6 good inversions in the mouse+rat ancestor, and 2 good inversions in the cow+dog ancestor (Figure 2.7a). The further progression of the algorithm is shown in Figures 2.7c-f (only four iterations are required to build the phylogeny). The 4 "dotted" edges in Figure 2.7 do not correspond to any micro-inversions (zero-length edges that have to be contracted) and thus represent the same genomic architecture. Representing this ancestral architecture as a single vertex results in the final tree shown in Figure 2.8(left). The currently accepted phylogeny on the same species constructed from accordant subsets from [88, 115, 8] is presented in Figure 2.8(right).

While a number of edges in the reconstructed tree remain unresolved, our analysis provides a proof of principle that micro-inversions represent valuable characters for phylogeny reconstruction. For example, the mitochondrial data analysis in [3] places the

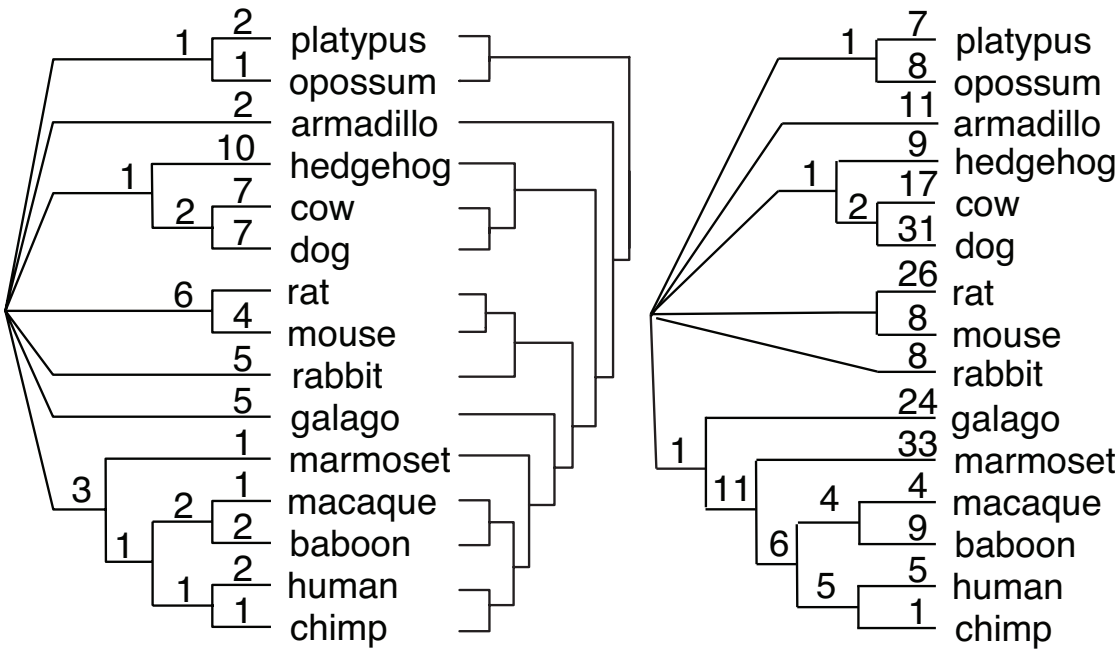


Figure 2.7 The reconstructed tree (*left*), the tree corresponding canonical mammalian phylogeny (center), and the tree that is reconstructed using 14 ENCODE targets (*right*). Vertices connected by dotted edges in Figure 2.8 (no corresponding micro-inversions) are contracted into a single vertex.

hedgehog close to the root of the placentals, while others argue against this [145]. Our result supports grouping of hedgehog with cow and dog, a result that is supported by most recent studies.

2.B.4 Micro-inversions in human and chimpanzee lineages

The problem of discovering micro-inversions requires a careful analysis even when comparing the human and chimpanzee genomes, where very high sequence similarity suggests that micro-inversion breakpoints should be easy to detect. We analyzed the 1460 putative micro-inversions reported in [42] that are shorter than 15kb, by running *InvChecker* on each inverted locus and 60kb of flanking sequence. Only 293 putative micro-inversions were classified as inversions by *InvChecker* while 1005 inversions were classified as artifacts. The remaining 162 putative micro-inversions represent ambiguous genomic architectures that *InvChecker* is unable to call either way (as demonstrated in Figure 2.9). A large fraction of these artifacts are palindrome-like structures. Feuk et

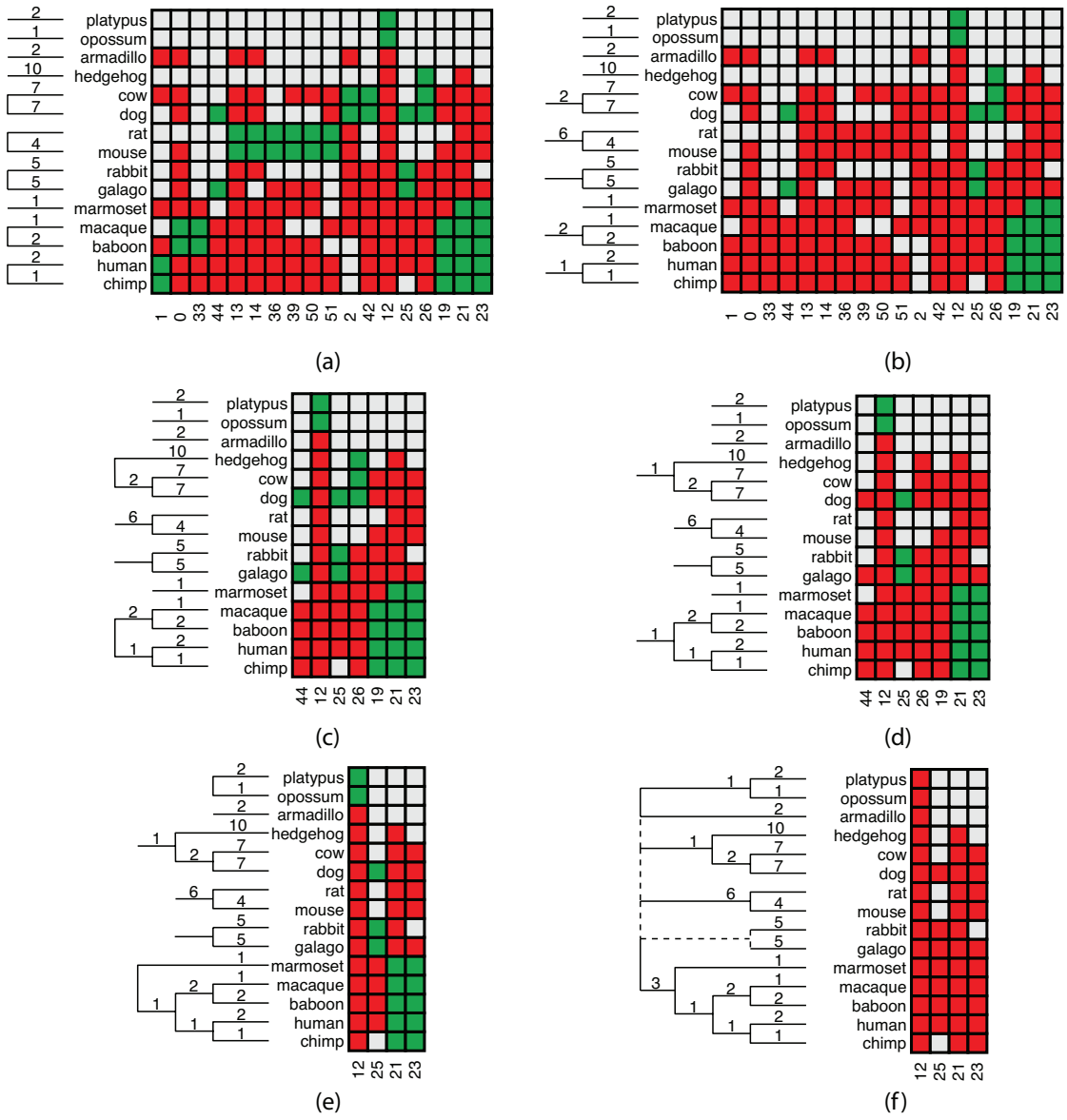


Figure 2.8 Progression of the tree reconstruction algorithm.

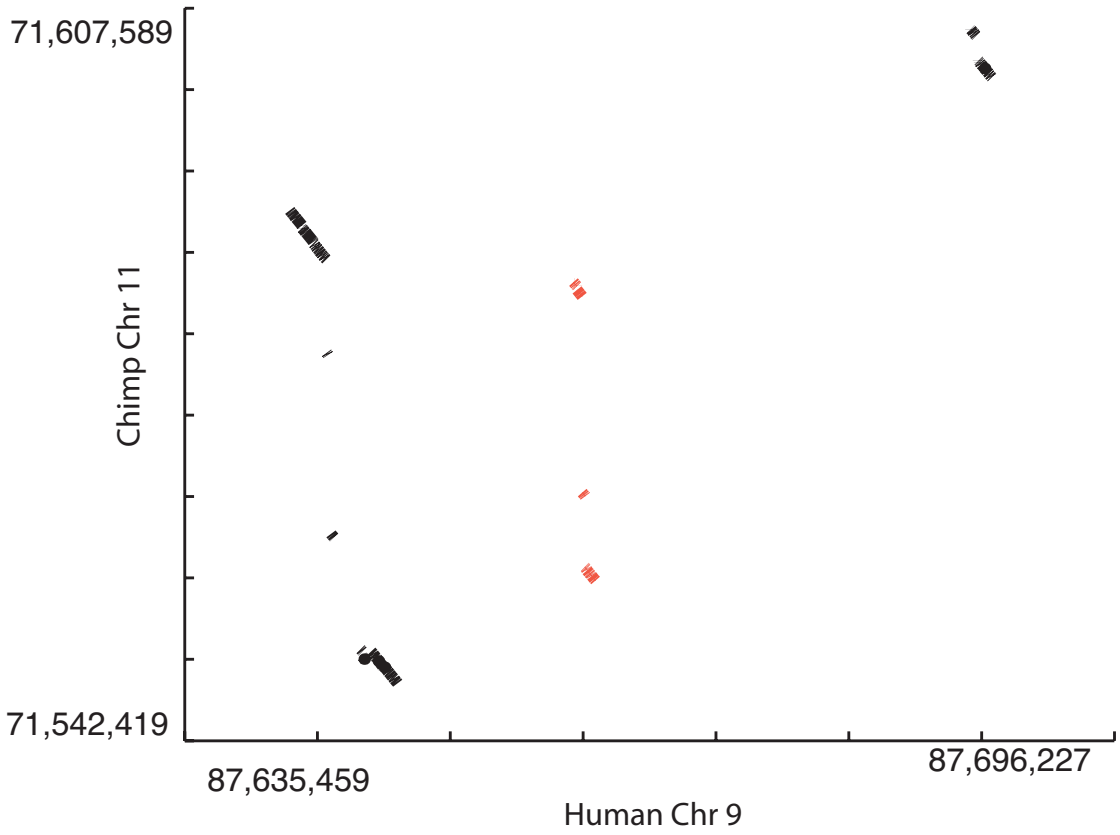


Figure 2.9 An inversion retained in the human/chimpanzee whole genome alignment for which *InvChecker* is unable to determine if it is a proper inversion. The netted inversion is in red. Because there is no surrounding forward alignment (within 60 kb region), we cannot determine if this inversion is along the main forward diagonal.

al, 2005 [42] experimentally validated some selected micro-inversions and confirmed that they indeed represent inverted sequences. Since a large portion of inversions in [42] represent artifacts, the questions arises how these artifacts can possibly be experimentally validated. Of the 19 experimentally validated inversions from [42] that were shorter than 15kb, *InvChecker* classified all of them as inversions except one (of length 4331 on chr 7) that turned out to be an inverted duplication. This suggests that the selection of inversions in [42] for experimental validation had a bias for selecting canonical inversions (like the first inversion in Figure 2.1). This bias is likely to be a consequence of the difficult repetitive nature of some breakpoint regions that makes PCR-based validation difficult.

We manually analyzed the genomic micro-architecture of these inversions using

genomic dot-plots of each inversion and flanking sequences. While a large number of microinversions are flanked by inverted repeats (34%), many are flanked by insertions (31%) (Appendix C). This suggests that while inversions are thought to arise from non-homologous recombination of inverted repeats, local genomic architecture is subject to rearrangement during repair of an insertion (or deletion).

2.B.5 Reconstruction on additional sequence

We applied *InvChecker* to two additional data sets: sequences of 14 species across the 14 ENCODE manually selected target regions (ENCODE), and sequences of 38 species released as part of the NISC Comparative Vertebrate Sequencing project (CVSP). The ENCODE data set yielded 120 inversion loci that produce a phylogeny shown in Appendix D. The additional regions added the resolution of the laurasiatheria and euarchontogilres clades as well as further support for previously defined clades. The CVSP data set yielded 122 characters that correspond to internal edges on the phylogeny, and produced a phylogeny shown in Appendix E. On average, roughly 3 Mb are sequenced in each species. Our phylogeny supports a Rodentia topology that differs with recent nuclear gene-based phylogenies [90], and is discussed in Appendix F. This data set also highlights the rare but necessary need to be able to detect overlapping micro-inversions, as we found an instance of this in dog and ferret (discussed in Appendix G).

2.C Discussion

We have presented a method to discover micro-inversions and to use them as evolutionary characters to reconstruct phylogeny. The reconstructed tree has a number of unresolved branches but otherwise is in agreement with the currently accepted phylogeny.

High sequence divergence in non-coding regions makes it more difficult to find micro-inversions at the deep branches of the mammalian evolution. In particular, we did not detect ancient inversions shared by a human/rodentia (Euarchontogilres) ancestor versus dog/cow/hedgehog (Laurasiatheria) ancestor. This is consistent with findings of Bashir et al, 2005 [8], who only found two repeats in the CFTR greater region separating Euarchontogilres from Laurasiatheria, compared with 112 that resolve primates. We were

able to reconstruct the primate phylogeny with rather short branches thus indicating that our limited capacity in finding "ancient micro-rearrangements" may reflect limitations of our inversion detection tool and the choice of stringent parameters rather than the shortage of ancient micro-inversions. Indeed, throughout this study we use the the same BLASTZ alignment scoring matrix and stringent $k = 2$ threshold for detecting micro-inversions, i.e., we apply the same threshold to human-chimpanzee comparison (high similarity) as to human-platypus comparison (low similarity). Making a variable threshold (depending on divergence of the species) will likely lead to significant decrease in the number of unresolved characters and discovery of ancient micro-inversions that remain undetected under the stringent threshold. Improving our algorithm to detect ancient micro-inversions remains an important goal that we are now addressing using ancestral sequence reconstructions [12].

We also remark that the assembly of most of 15 genomes we consider are still incomplete and are currently represented by multiple contigs. These contigs are mapped to human genome to produce ordered sequences. This procedure imposes a "human order" on unfinished sequences and prevents us from detecting inverted sequences in some species even if they exist. Such sequences may be discovered with completing the NISC project. In the future, ongoing genome sequencing projects (even low coverage) will enable micro-inversion based phylogenomics.

2.D Acknowledgments

We are grateful to Ali Bashir, Vineet Bafna, Bill Murphy, Stephen Scherer, and Glenn Tesler for many useful comments. A few days before this paper was submitted we learned that Jian Ma and Jim Kent (personal communication) came up with an approach that finds micro-inversions by post-processing of netted alignments and arrived to a similar estimate of the number of micro-inversions between human and chimpanzee (approximately 500). We are grateful to Jian Ma and Jim Kent for the comparative analysis of human-chimpanzee micro-inversions found in this paper and many useful comments. Some of the utilized sequence data were generated by the NIH Intramural Sequencing Center (www.nisc.nih.gov). BJR holds a Career Award at the Scientific

Interface from the Burroughs Welcome Fund.

2.E Appendix A

Discovery and Validation of Micro-Inversions

We analyzed pairwise alignments with BLASTZ [130], axtChain, chainNet [68], and our program, InvChecker. For completeness, we include a more detailed description of the chaining and netting algorithms, as described in [68]. The output of BLASTZ is a set of high-scoring local alignments between a reference and query sequence. A chain of local alignments, or chain is a non-overlapping, maximally scoring subset of the local alignments that has consistent order in each species (no rearrangements are allowed). Thus, if there is a rearrangement, such as an inversion, there will be a gap in the chain starting and ending at the breakpoints of the inversion. A net of chained alignments, or a net, is a set of full or partial chains that attempts to assign every nucleotide in one sequence an alignment from a chain. The set is initialized to the highest scoring chain, and agglomerates non-overlapping portions of the next highest scoring chain until as many nucleotides as possible in one sequence are assigned in an alignment. Thus if there is an inversion between two sequences, the net will comprise one chain that has alignments of the forward strand of each sequence, with a gap in the alignments over the inversion, and another chain that has alignments of the reverse strand of the query. An inverted duplication in the reference sequence would similarly be stored as a net with two chains, although the region in the query sequence that aligns to the duplication would be involved in two chained alignments. We do not use chaining and netting to detect inversions, but only to assemble BLASTZ hits into a main diagonal. We then considered remaining reverse strand alignments in the context of the main (direct) diagonal to detect inversions[4]. Insertions of repeats and sequence divergence often split continuous inverted regions into disjoint segments thus creating several BLASTZ alignments instead of a single one (Figure 2.11). We assemble such disjoint segments into a single region by combining compatible reverse-strand alignments (regardless whether or not they were chained by axtChain).

[4]Below, micro-inversions are referred to as inversions for brevity.

Our approach for finding inversions is motivated by a simple observation that captures the differences between true inversions (three top diagrams in Figure 2.10a, in supplementary information) and artifacts (four bottom diagrams in Figure 2.10a). If we invert the reversed (anti-diagonal) segments in the top three diagrams, the score of the direct strand alignment will increase resulting in a somewhat "longer" direct diagonal (Figure 2.10b, upper). However, if we invert the anti-diagonal segments in the lower four diagrams, the score of the direct strand alignment will unlikely to increase since inversions of the anti-diagonal segments do not contribute to the main diagonal (see Figure 2.10b).

Let ρ be a substring of sequence P and $\rho \cdot P$ be a sequence obtained from P by inversion of the substring ρ. Given a sequence Q, we define $\Delta(\rho)$ as $s(\rho \cdot P, Q) - s(P, Q)$, where $s(*, *)$ is the score of alignment between two sequences. $\Delta(\rho)$ represents the gain in the alignment score after inverting substring ρ in P. Given sequences P and Q and a parameter τ we are interested in all substrings of P with $\Delta(\rho) > \tau$, i.e., all inversions with significant gain in alignment score.

Although it is possible to rigorously incorporate inversion search into the formulation of the classical global alignment problem [128, 18], the problem requires $O(n^4)$ time for sequences of length n and becomes computationally prohibitive in the rigorous setting. Schoniger and Waterman [128] described a heuristic approach based on finding only top-scoring suboptimal reverse strand local alignments and rigorously combining them with all direct strand alignments. We follow this approach by considering only substrings ρ aligned in statistically significant reverse strand local alignments and analyzing whether they represent true inversions so that $\Delta(\rho) > \tau$.

To simplify the analysis of micro-inversions, a 50 kb inverted sequence in macaque was manually inverted back to exclude nested inversions from consideration[5].

[5]The nested micro-inversions require a careful analysis since the relative ordering of larger and smaller (nested) inversions is unknown. However, in case of this inversion, the analysis of other close species (e.g. baboon) confirmed that the larger inversions happened after the shorter inversions thus justifying this manual procedure.

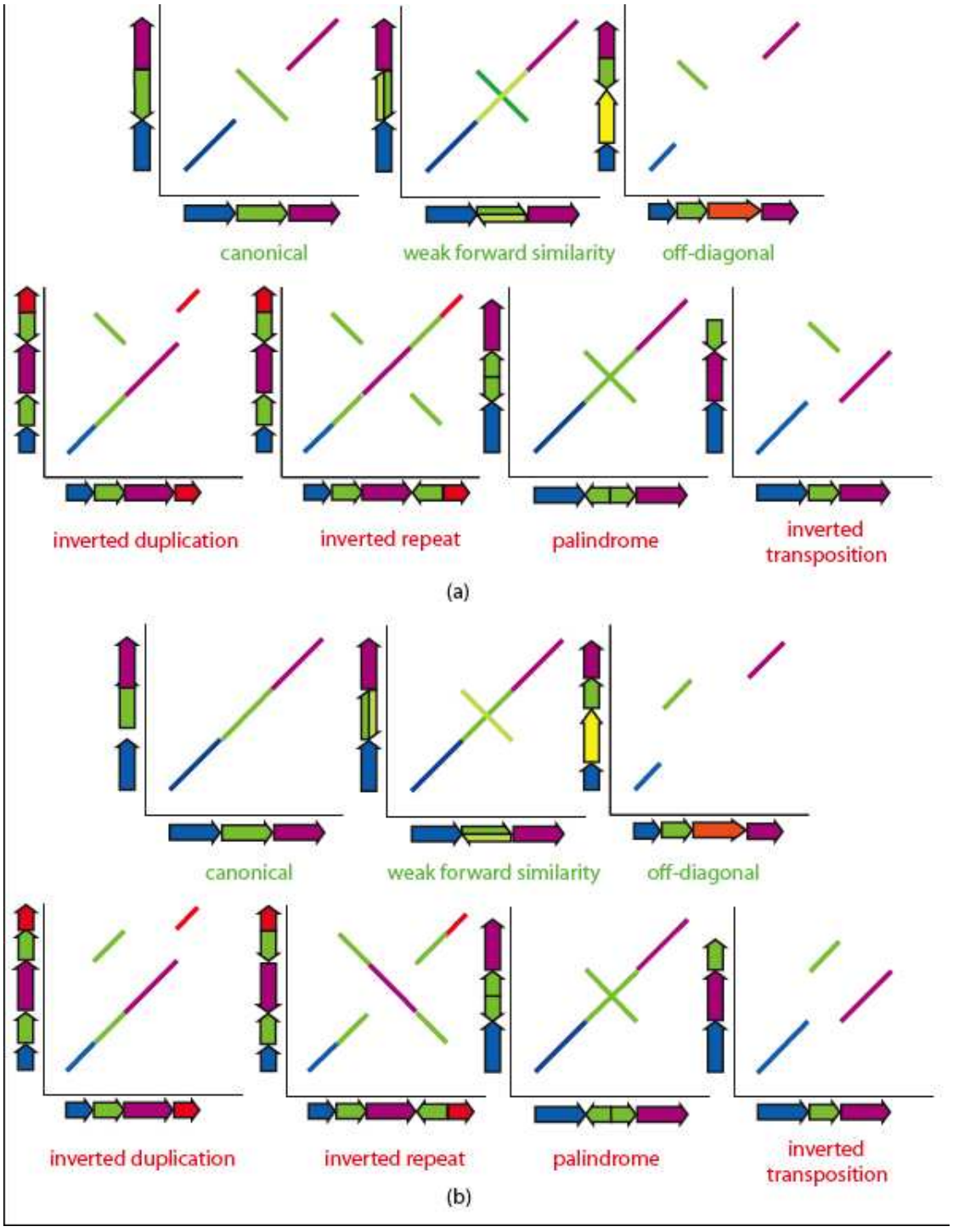


Figure 2.10 (a) Diagrammatic dot plots, reproduced from the main paper. (b) The result of reverting the inverted segemnts, to visualize how alignment score could increase after inverting a segment (without transposition).

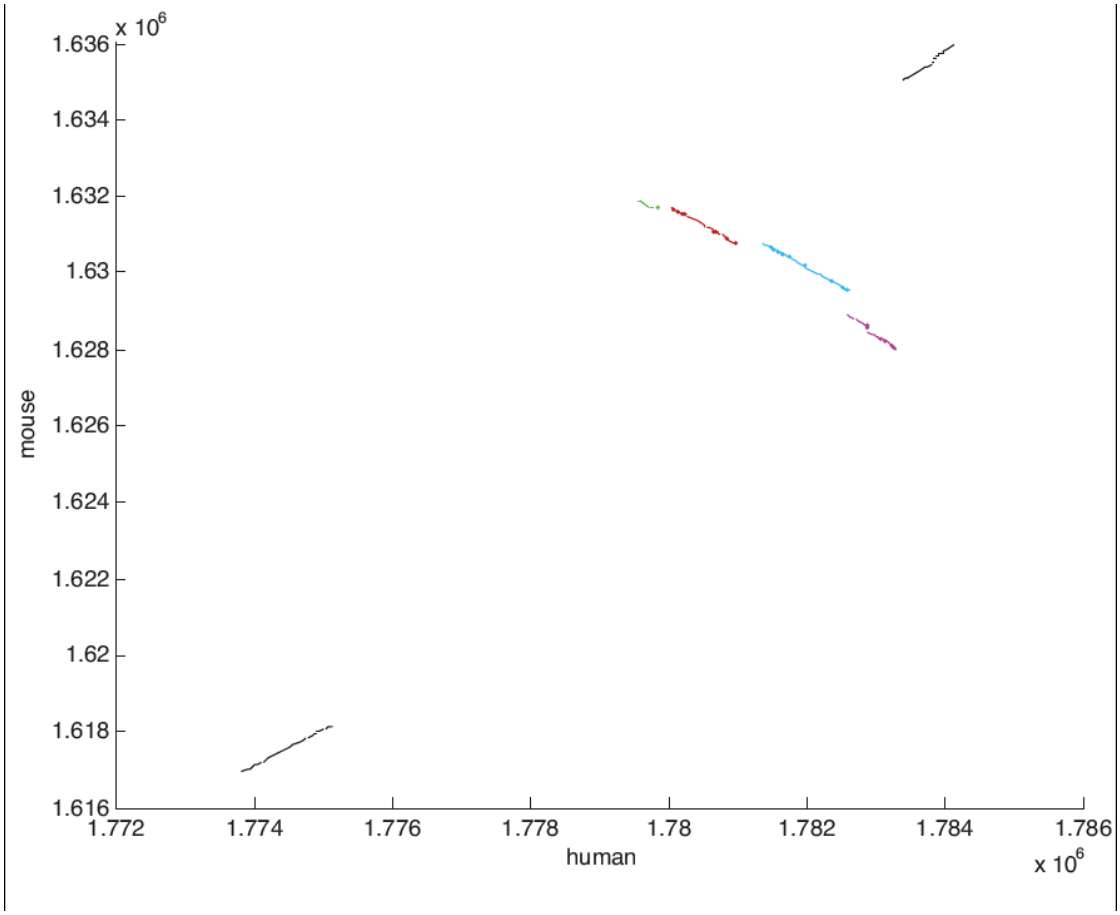


Figure 2.11 A single, divergent inversion that has been split into several different alignments (colors) by BLASTZ.

2.F Appendix B

Distribution of Micro-Inversions Along the Genome

We examined the inversion loci in the human CFTR region to analyze their distribution with respect to exons. 55 out of 67 inversion loci are present in the human CFTR region, while the remaining inversion loci were either deleted in the course of evolution or too divergent to be detected in the human genome. However, the positions of most of these inversions can still be projected to the human genome by mapping their flanking regions (Figure 2.12).

We hypothesize that there is evolutionary pressure against inverting exons (and possibly some regulatory regions), but that inversions are distributed somewhat randomly along the CFTR region. Indeed, no inversions overlap the exons of human genes. Moreover the closest micro-inversion to the gene start (unique to galago) is located 670 nucleotides upstream of the CFTR gene. Using the start and end coordinates of human genes, we found that 36 inversions map to transcribed sequence in the human genome.

Assuming a random distribution of inversion loci one would expect 34.7 inversions to lie within transcribed sequences, consistent with the observed number and pointing to a somewhat random distribution of inversion loci in non-coding regions. The minimum distance between any of the 55 inversions mapped to human is estimated as 0.5 kb.

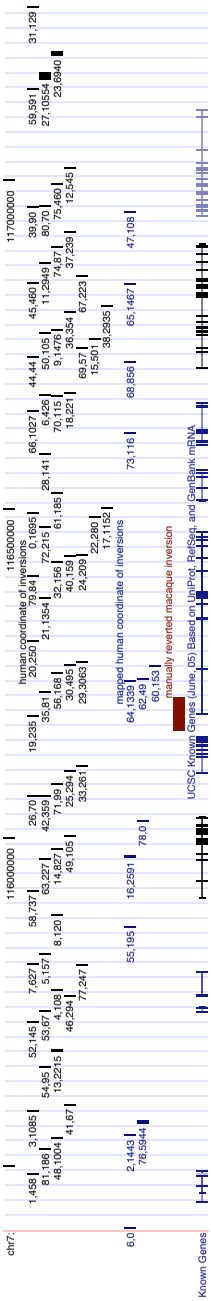


Figure 2.12 The distribution of inversions in the human CFTR region. Each inversion is labeled by a pair (a,b), where 'a' is the index of the character used in reconstruction, and 'b' is the length of the inversion.

2.G Appendix C

Phylogeny of Fourteen Species on an Extended Set of ENCODE Targets

The ENCODE dataset consists of two separate classes of targets, manually selected genomic regions, and stratified random regions of varying levels of conservation. We elected to reconstruct phylogeny using the manually selected regions as they are high in gene content and less likely to contain complicated rearrangements that our method currently does not take advantage of.

The September 2005 release of sequences for ENCODE manually selected regions, ENm001-ENm014, were downloaded from the UCSC ftp site: `ftp://hgdownload.cse.ucsc.edu /goldenPath/hg17/encode/alignments/SEP-2005/sequences`. Repeat libraries were generated with RepeatScout [113], and masked using both RepeatScout and RepBase [65] repeat libraries using RepeatMasker. Any sequence with more than 20 contigs was discarded. We ran InvChecker on the pairwise alignments of repeat-masked sequences. We removed 6 characters that cuase four-gamete violations using Maximum Conflict Removal [8].

The phylogeny (Figure 2.13) is consistent with the one generated on the CFTR target, ENm001, an constains more support for each edge. There is an additional inversion that separates laurasiatheria from euarchontogilres. The data that was used to generate the tree is shown in Figure 2.14.

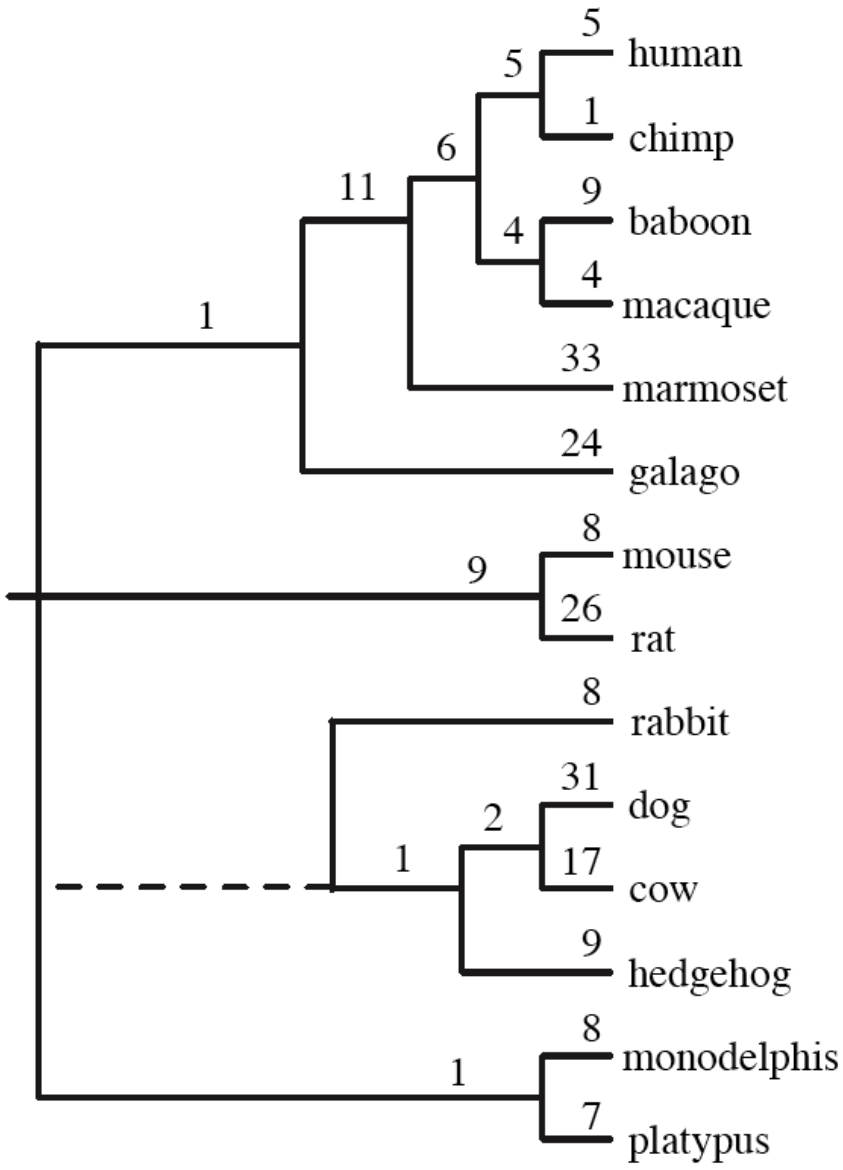


Figure 2.13 The phylogeny of 14 species generated using characters from the 14 ENCODE manually chosen targets. The extended data set found more species-unique inversions as well as additional support for all internal edges except for the encompassing clades for galago, rabbit, and hedgehog.

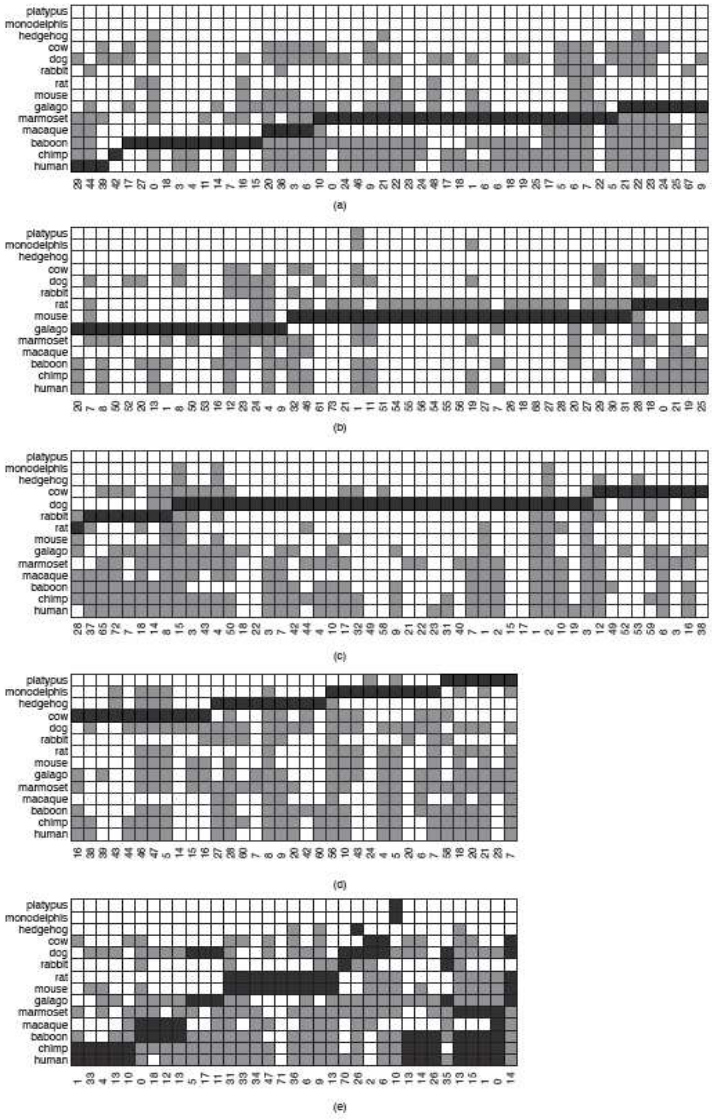


Figure 2.14 The steps for reconstructing phylogeny on14 species generated using characters from the 14 ENCODE manually chosen targets.

2.H Appendix D

A comparison of Human-Chimpanzee inversions identified by Ma et al [77]. and Chaisson et al [25].

Recent studies by the UCSC-PSU group (Ma et al) and the UCSD group (Chaisson et al.) developed new algorithms for detecting micro-inversions and came up with similar estimates for the number of human-chimpanzee micro-inversions (428 and 426). These estimates are significantly lower than the recent estimate in Feuk et al., 2005 [42], and moreover, many micro-inversions found in these papers were not discovered in [42]. While Ma et al. and Chaisson et al. agree on 251 micro-inversions, 175 micro-inversions in Ma et al. were not found by Chaisson et al. and 175 (coincidentally) micro-inversions in Chaisson et al. were not found by Ma et al. The discrepancy between the total (428 = 251+175+2) is accounted for by inversions that were fragmented and treated separately in one study. The disagreement between two studies emphasizes that finding micro-inversions is a difficult problem even in a seemingly trivial case of human-chimpanzee comparison where the similarity between two sequences is very high. We therefore decided to manually validate all annotations in Ma et al. and Chaisson et al. in a case-by-case fashion to better understand the reason why two software tools sometimes disagree in classifying micro-inversions.

We found that although both approaches are complementary, the method of Chaisson et al. is slightly more sensitive than the method of Ma et al. In particular, the false positive rate of Chaisson et al. is 17% while the false positive rate of Ma et al., is 23%. The main source of false positives in both studies are palindromes (inverted duplications) in human that undergone partial deletions in chimpanzee (and vice versa). Inversions that are found by Ma et al. that were missed by Chaisson et al. were generally flanked by inverted repeats, and were difficult to discern as inversions or inverted repeats. We are currently addressing this complication in a new release of InvChecker. Furthermore, the study by Chaisson et al. aimed to gain an estimate of the total number of inversions in human and chimpanzee, and examined only roughly 2.6 Gb of genomic sequence.

Since 80% of micro-inversions found in [42] represent artifacts, their main con-

clusions about the interplay between micro-inversions and repeats have to be revisited. We now estimate that 34% of all human-chimpanzee micro-inversions are flanked by inverted repeats. These repeats are likely to promote micro-inversions. We also found that a surprisingly large number of micro-inversions (31%) are flanked by insertions rather than by repeats (the interplay between micro-inversions and insertions was overlooked in [42]). Another surprising conclusion is the discovery of a large number of cryptic micro-inversions whose flanking regions do not exhibit any detectable similarity between human and chimpanzee genomes. These flanking regions without detectable similarity are often long, a surprising finding since human and chimpanzee genomes are very similar. We point to the parallel between this micro-rearrangement phenomenon of "lost similarity" and the breakpoint regions in the context of the whole genome comparison that also often do not exhibit any detectable sequence similarity.

Below is a summary of a manual case-by-case comparison of micro-inversions that were found between human, release hg17 and chimpanzee, release ptr1. Dot-plots were generated (with exact matches of length 12) of sequences 21 times the length of each inversion, with the inverted sequence in the middle.

Dot plots of inversions found by Ma et al. are at:

`http://bioinf.ucsd.edu/~mchaisso/jian/coords10/index.html`

And plots of inversions found by Chaisson et al. are at:

`http://bioinf.ucsd.edu/~mchaisso/mark/coords10/index.html`

We then manually classified found micro-inversions into four classes (Figure 2.15):

1. Canonical inversion
2. Miscalled inversion.
3. Cryptic inversion
   Type 1 (one of the flanking direct
   strand alignments is missing)
4. Cryptic inversion
   Type 2 (both of the flanking direct
   strand alignments are missing).
5. Others (transpositions, nested inversions, and multiple
   rearrangements).

We describe each type of inversion in further detail. An example of a dot-plot of a canonical inversion is in Figure 2.15a. There are neither horizontal or vertical gaps nor overlaps of the diagonals with the anti-diagonal, indicating that a unique sequence

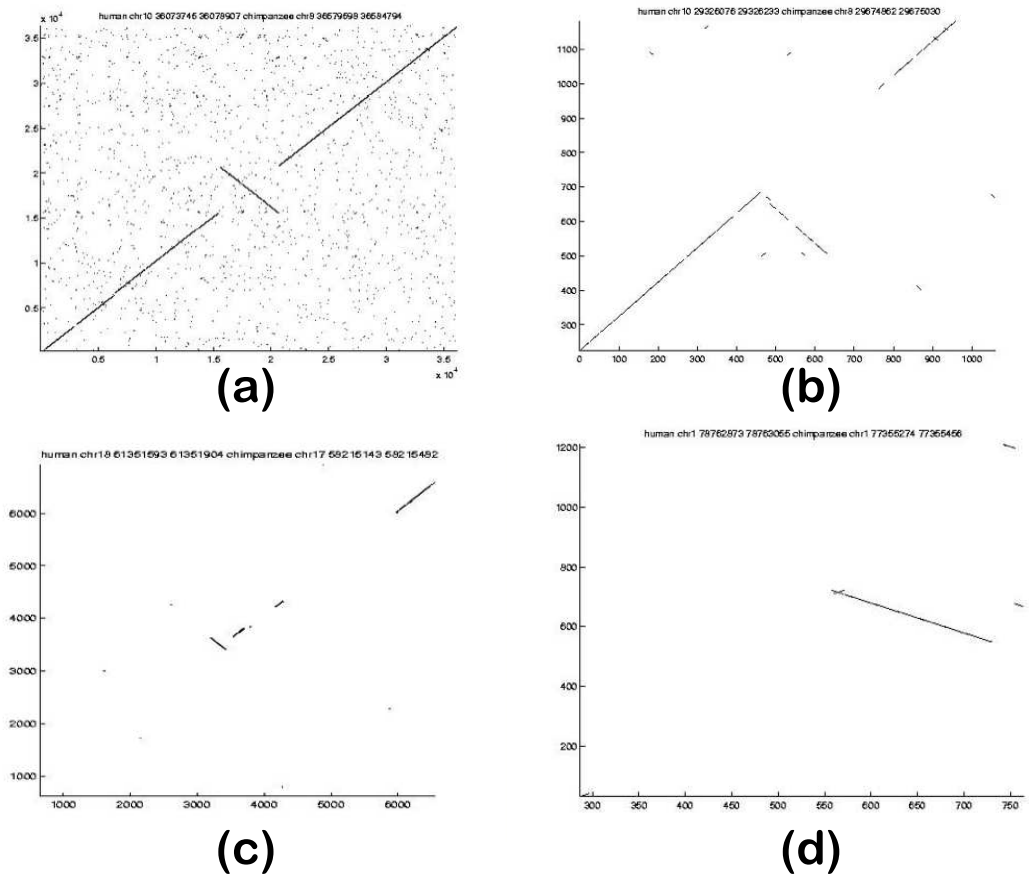


Figure 2.15 Four examples of dot-plots. (a) a canonical inversion, (b) a misclassified inversion, (c) a cryptic single diagonal and single anti-diagonal, and (d) a cryptic anti-diagonal.

has been inverted. We show an anti-diagonal that has been misclassified as an inversion in Figure 2.15b. Such anti-diagonals represent reverse-duplicated features of a single genome.

There are many inversions that show up, 'cryptically' in dot-plots as only a single anti-diagonal segment (Figure 2.15c) that are not flanked by diagonal segments on both the left and right, as opposed to the previous plot that has two diagonal segments. While this may reflect the limitations of parameter choice for analyzing flanking regions, it also begs the question as to why the flanking diagonal segments are not detected in the dot-plot in sequences so close as human and chimpanzee. Two possibilities are that there is a large insertion in one sequence, or there is a gap in the assembly shortly before the inversion in one species. Misassembled contigs may mimic inversions, and we propose that such dot-plots are of sequences that are more likely to have been misassembled, rather than inverted and simultaneously diverged, and thus account for them differently. Also, often the ends of supercontigs are enriched for assembly errors, and such cryptic dot-plots are more likely to be artifacts from assembly. Also, there exist cryptic inversions that only appear as a single anti-diagonal in the dot-plot, as shown in Figure 2.15d. This is a more extreme case of the previous example, and as such is not counted as an inversion. Finally, there are other cases that are likely to be nested or overlapping inversions, as shown in Figure 2.16, and these are not counted as inversions.

Two other interesting genomic architectures were also documented (Figure 2.17):

1. Flanking inverted repeats.
2. Flanking insertions.

For inverted sequences, we also counted how often they were flanked by either an insertion or flanking inverted repeats. Such tallies help give insight to the mechanism of how sequences are inverted. The summary is given in Table 2.1.

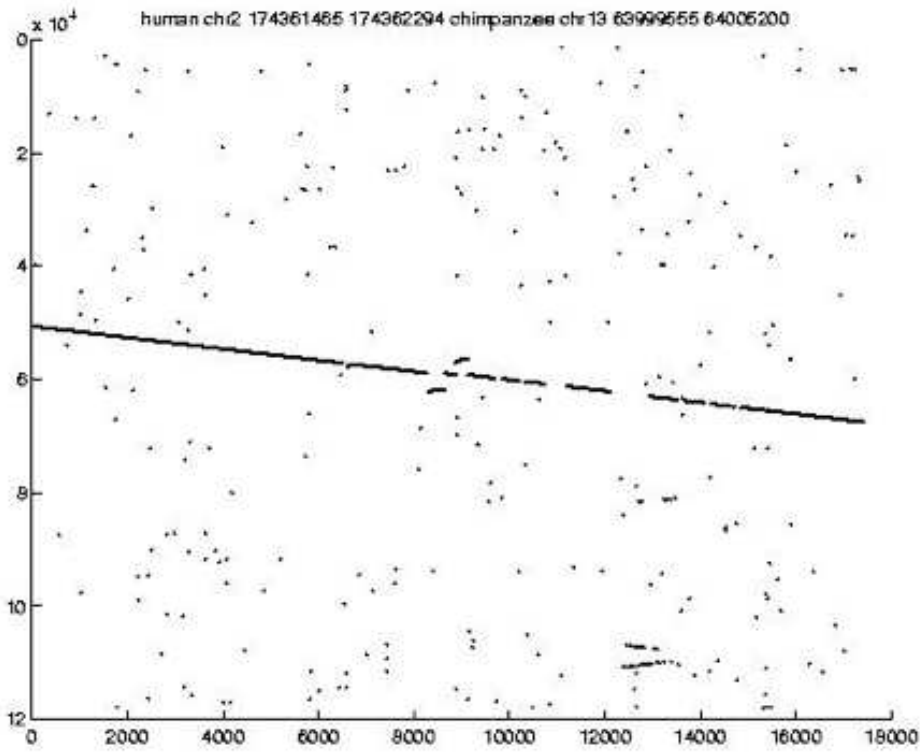


Figure 2.16 An example of nested inversions.

Table: 2.1 Counts of the various types of inversions classified using dot-plots of sequences in [77] and [25]

M: Ma et al
C: Chaisson et al.

	M	C	M ⋂ C	M ⋃ C
Total sequences	428	426	246	606
Canonical inversions	248	296	200	344
Flanking inverted repeats	43	100	42	101
Flanking insertions	82	93	82	93
Miscalled sequences	101	73	12	172
Cryptic inversion, Type 2	36	27	12	51
Cryptic inversion, Type 3	28	26	20	34
Others	15	4	4	15

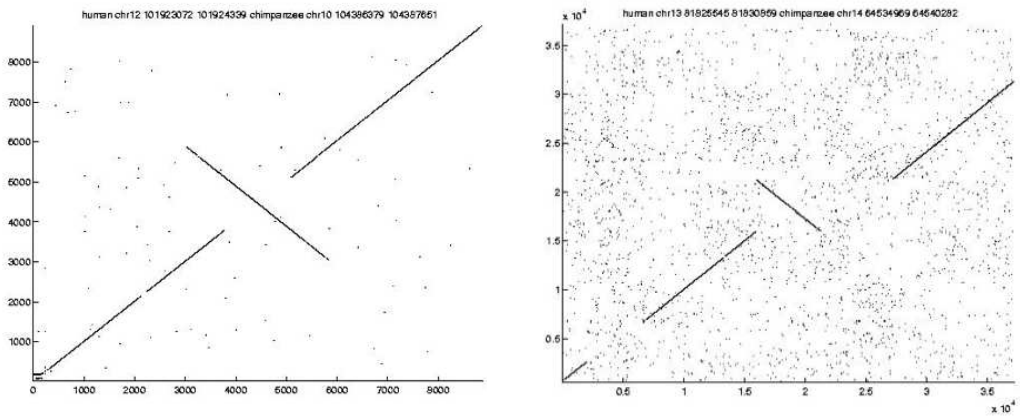


Figure 2.17 Examples of inversions that are accompanied by other genomic features. On the left, the inversion is flanked by inverted repeats. On the right an insertion/deletion is immediately adjacent to the inversion.

2.I Appendix E

The BAC sequences for the most well-covered targets (1, 3, 5, 6, and 13) were downloaded from Genbank and joined together using our own method similar to the TPF processor (`http://www.ncbi.nlm.nih.gov/projects/zoo_seq/`). We generated repeat libraries for each species using RepeatScout [113] and RepBase [65] libraries and masked each genome with sensitive masking using RepeatMasker. Since low copy-number repeats may lead to false positive repeat detection, we additionally masked each sequence by performing a self-alignment using BLASTZ, and masking all sequences involved in multiple alignments. Inversions were detected using InvChecker on pairwise alignments of the repeat-masked sequences, as before. Figure 2.18 shows inversion loci where more than one species is found in each orientation.

The sequences within each distinct targeted region are orthologous and may be used to detect inversions. However, the NISC Comparative Vertebrate Sequencing Project is ongoing, and the number of species sequenced in each target varies widely. Table 2.2 presents a summary of the amount of sequence available for each species. Most species are either sequenced in just one target (target 1), or are sequenced in almost every target, such as species whose entire genomes have been sequenced. Adding additional targets for species in the second category is not likely to aid in the current study, since

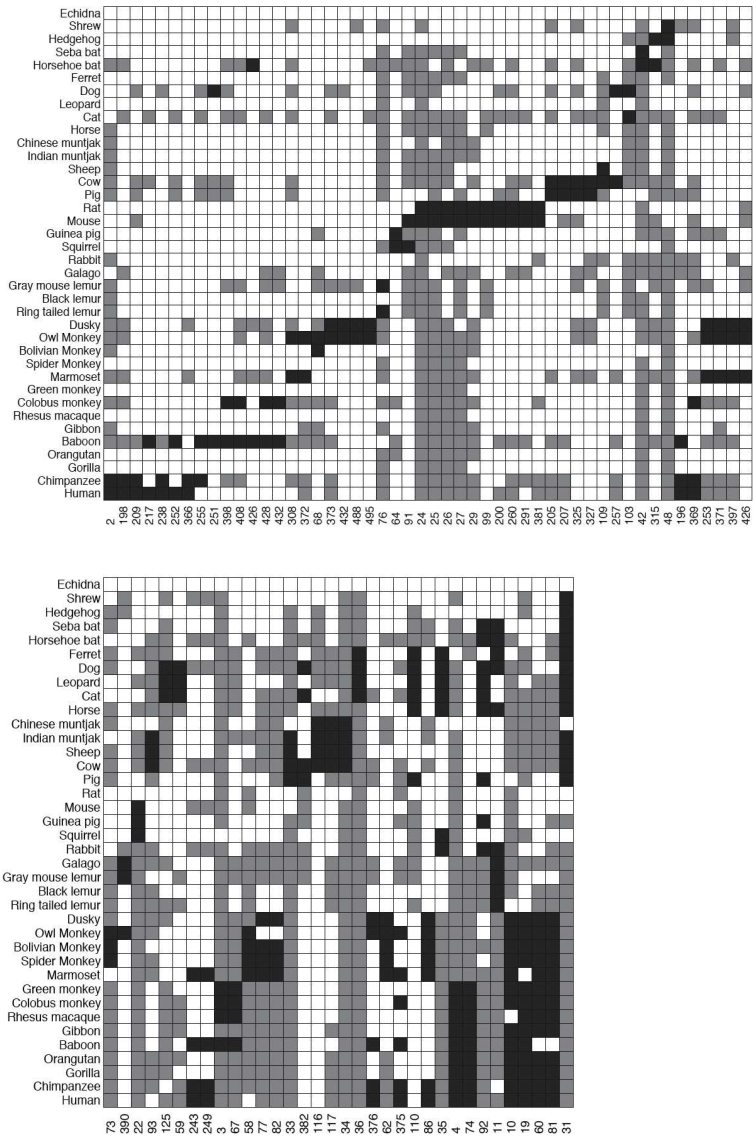


Figure 2.18 Inversions in the ancestor of at least three to fourteen extant species.

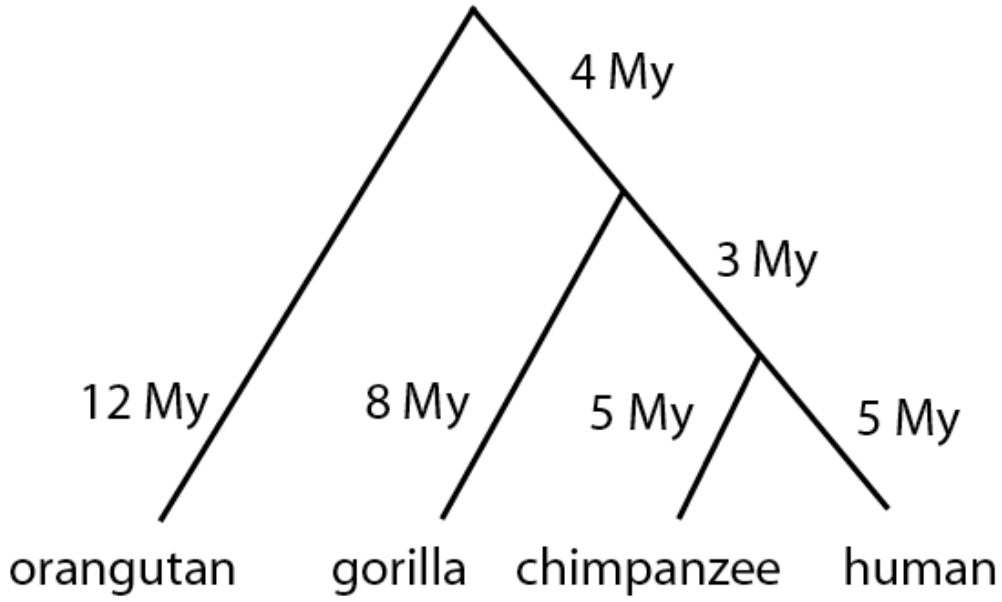


Figure 2.19 Four-primate phylogeny with an estimate of the number of years of evolution along each branch.

all of the unresolved clades, such as the (human, chimpanzee, gorilla, orangutan) clade only contain two species that have extra sequenced targets (in this case human and chimpanzee), and characters with known values for all four species are required to resolve the clade.

We note that with the inversion rate that we have observed: one inversion per 66 million years (My) of evolution per million bases (Mb), it is not surprising that additional targets would be required to resolve some clades, such as the (human, chimpanzee, gorilla, orangutan) clade. According to the accepted topology of this clade (Figure 2.19), a total of 7My 6Mb = 42 My Mb of evolution have passed on the in internal edges of the phylogeny, so less than one inversion is expected between ancestral species using target 1 alone. However, because there are many hundred inversions that have arisen between human and chimpanzee in 5 million years, when full genome sequences are present there will be ample information to resolve all primate phylogenies.

The phylogeny is shown in Figure 2.20. The phylogeny is consistent with that observed in other studies [90], although there are a few unresolved clades. The only dis-

crepancies with the tree presented in [90] are discussed in further detail in SI Appendices F and G. Two of the interesting features of this tree are that there are several species that have idential genomic architecture, such as gorilla and orangutan, while other species that have an extremely high number of inversions, such as galago, hedgehog, cat, and dog. Most of the species that are have a high number of inversions are sequenced in all 5 targets included in this study. The CFTR greater region is gene rich (10 genes in human), and may be under greater evolutionary pressure against micro-rearrangements. This may provide an explanation for the identical genomic architectures that are observed.

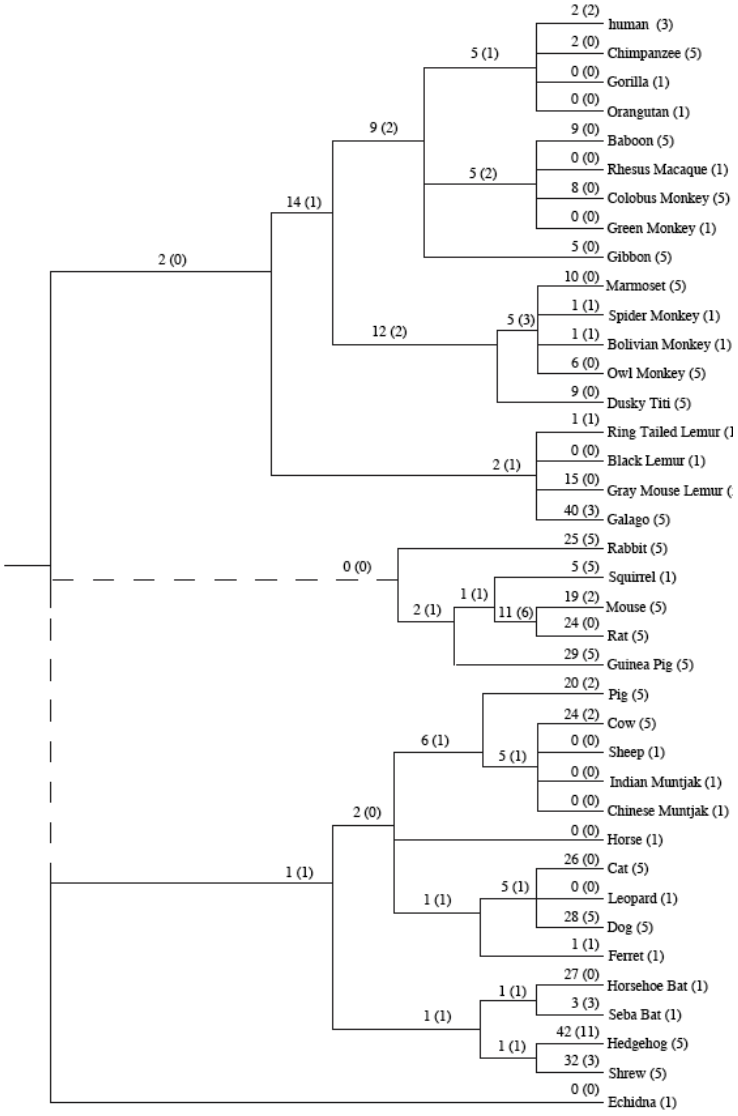


Figure 2.20 Mammalian inversion-based phylogeny. Each edge is labeled with the number of inversions that may be reversed in a species (extant or ancestral) in order to make its genomic architecture closer to all other species, and in parenthesis is the number of such inversions that are found in target 1. To the right of each species the number of targets used in the study is listed in parenthesis.

Table: 2.2 Summary of sequences for each species. The aggregate number of masked, unmasked, unknown, and total sequence length from all targets available for each species islisted, along with the number of targets used for each species. M, megabases; k, kilobases.

Species	Num Repeat--masked	Num. non-masked	Unknown	Total	Num. targets
Human	2.01 M	2.56 M	0	4.57 M	3
Chimpanzee	5.02 M	6.54 M	354.42 k	11.92 M	5
Gorilla	707.22 k	1.07 M	4.05 k	1.78 M	1
Orangutan	732.51 k	1.07 M	3.10 k	1.80 M	1
Baboon	5.92 M	7.47 M	21.20 k	13.42 M	5
Gibbon	1.75 M	2.25 M	7.04 k	4.01 M	5
Rhesus	640.95 k	1.03 M	0	1.67 M	1
Colobus	3.06 M	3.86 M	10.31 k	6.94 M	1
Green monkey	713.47 k	1.11 M	250	1.82 M	1
Marmoset	4.56 M	5.93 M	26.20 k	10.52 M	5
Spider monkey	692.47 k	1.03 M	4.15 k	1.72 M	1
Bolivian monkey	895.90 k	1.31 M	800	2.21 M	1
Owl monkey	2.99 M	4.11 M	14.46 k	7.12 M	5
Dusky titi	2.97 M	4.11 M	14.05 k	7.10 M	5
Ring-tailed lemur	417.29 k	983.73 k	250	1.40 M	1
Black lemur	417.41 k	952.37 k	1.8 k	1.37 M	1
Gray mouse lemur	2 .41 M	4.25 M	13.35 k	6.68 M	5
Galago	4.14 M	6.78 M	20.01 k	10.94 M	5
Rabbit	4.06 M	6.06 M	29.31 k	10.16 M	5
Squirrel	354.70 k	1.07 M	2.25 k	1.43 M	1
Guinea pig	1.92 M	4.75 M	9.70 k	6.68 M	5
Mouse	4.06 M	9.38 M	147.47 k	13.59 M	5
Rat	3.01 M	7.30 M	247.02 k	10.56 M	5
Pig	3.91 M	6.25 M	21.34 k	10.19 M	5
Cow	4.86 M	6.74 M	26.02 k	11.63 M	5
Sheep	826.98 k	1.19 M	3.41 k	2.02 M	1
Indian muntjak	747.67 k	1.16 M	3.05 k	1.91 M	1
Chinese muntjak	569.66 k	885.74 k	2.70 k	1.45 M	1
Horse	659.11 k	1.24 M	0	1.90 M	1

2.J Appendix F

Analysis of the Rodentia Clade

Table 2.2 *Continued.*

Species	Num Repeat--masked	Num. non-masked	Unknown	Total	Num. targets
Cat	4.26 M	7.10 M	27.12 k	11.39 M	5
Leopard	660.80 k	1.03 M	6.45 k	1.70 M	1
Dog	3.52 M	6.90 M	35.44 k	10.46 M	1
Ferret	535.75 k	955.53 k	5.35 k	1.49 M	1
Horsehoe bat	2.81 M	5.87 M	20.57 k	8.71 M	5
Seba bat	522.41 k	708.14 k	750	1.23 M	1
Hedgehog	3.70 M	3.82 M	23.04 k	7.55 M	5
Shrew	3.27 M	6.15 M	45.59 k	9.47 M	5
Echidna	260.22 k	448.29 k	6.31 k	714.84 k	1

In this appendix we present the analysis of part of the Rodentia clade: guinea pig, squirrel, and mouse. The phylogenetic position of guinea pig is a very controversial topic in systematic biology. For example , nuclear gene studies suggest guinea pig and mouse as sister species [90, 78], while mitochondrial studies suggest squirrel and guinea pig as sister species [116], or mouse and squirrel as sister species [75]. We show that there is an inversion that supports the phylogeny: (guinea pig (squirrel, mouse)). Since our phylogeny groups squirrel with mouse with only one inversion, we show a detailed description of this inversion by taking galago as an out-group, and displaying the alignments for the regions surrounding the inversion. We performed the alignments using BLASTZ; however, because the divergence in each species is quite high we used a lower HSP cutoff, K=1000 (defaultK=3000). In each alignment plot, we draw boxes to indicate the boundaries of the inversion locus in each species, and list the boundaries in Table 1.

We note that this inversion is in a region that has been subject to both much divergence and many insertions. This is exemplified in the lack of resolution of the dot-plot of guinea pig-squirrel, where there are no significant alignments in the region corresponding to the inversion locus, and in galago, which contains an insertion within the inversion locus. We assign an orientation of 1 to loci that are in the same orientation in Figures 2.21-2.23, 0 to species in the opposite orientation, and ? for unknown, as summarized in Table 2. Although there are unknown characters in the table, their orientation is not necessary to reconstruct the phylogeny because the only consistent orientation is 0. Because we found only one inversion that supports the phylogeny:(guinea pig (squirrel, mouse)) we do not present a definitive proof of a particular rodent phy-

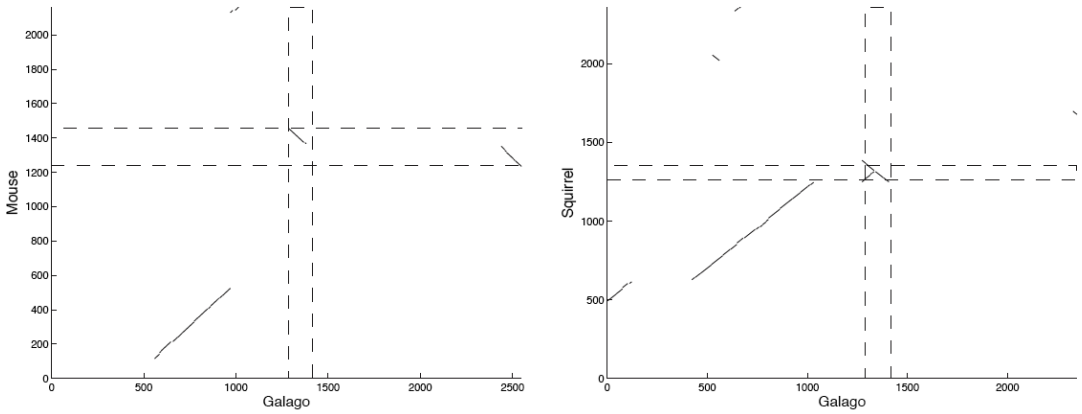


Figure 2.21 Galago/mouse and galago/squirrel alignments. The same region: 1287-1416 is inverted on the galago sequence between both mouse and squirrel.

Table: 2.3 Inversion locus boundaries. *Start, End*: Coordinates of each subsequence in target 1. *Locus start, Locus end*: Coordinates of the inversion locus for each species within the subsequence.

Species	Start	End	Locus start	Locus end
Galago	850200	852800	1287	1416
Squirrel	658400	661000	1260	1353
Guinea pig	754800	757400	1131	1386
Mouse	706000	708720	1353	1455

logeny, since it may be a rare example of homoplasy. Squirrel is sequenced only in target 1, thus presents limited sequence to detect microinversions. In particular, in our analyis of the NISC sequences (Appendix E) we detected 19 inversions in mouse, and 28 inversions in guinea pig, each of which is sequenced in five targets, and only one in squirrel. We hope that when all targets are sequenced for rodents our technique will provide a definitive answer to this ongoing debate of the phylogeny of guinea pig.

Guinea pig/mouse and guinea pig/squirrel alignments. The same region on mouse, 1353-1455, is inverted in guinea pig and galago. The squirrel/guinea-pig region is too divergent to align. The short forward diagonal in the region where squirrel and galago loci overlap is a weak forward similarity, and is not significantly high-scoring to consider an inverted duplication. The coordinates of the inversions are shown in Table 2.3 and the relative orientations are shown in Table tab:relinv.

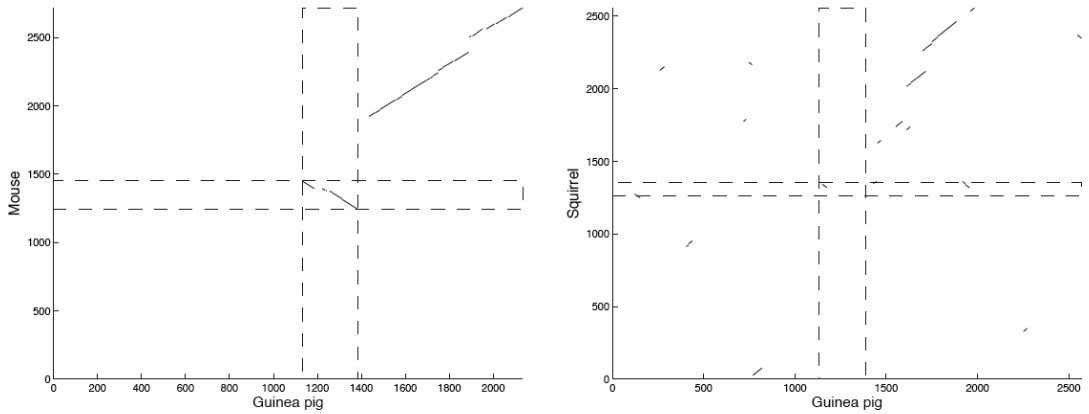


Figure 2.22 Guinea pig/mouse and guinea pig/squirrel alignments. The same region on mouse, 1353-1455, is inverted in guinea pig and galago. The squirrel/guinea-pig region is too divergent to align. The short forward diagonal in the region where squirrel and galago loci overlap is a weak forward similarity, and is not significantly high-scoring to consider an inverted duplication.

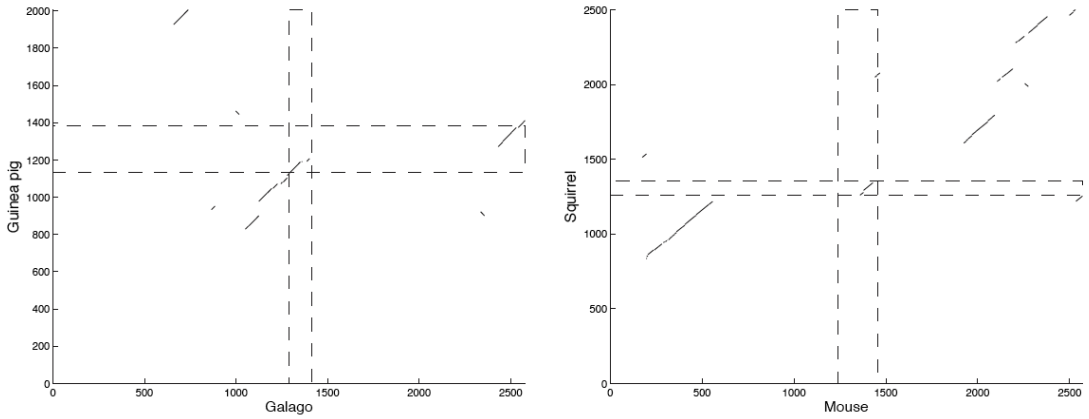


Figure 2.23 Galago/guinea pig and mouse/squirrel alignments. Each pair of genomes is colinear, confirming the orientations of the inverted regions within the other pairwise alignments.

Table: 2.4 Orientations of the inversion locus separating guinea pig from mouse and squirrel. Loci that are labeled with 1 are in the same orientation, 0 opposite, and ? unknown.

Species	Mouse	Squirrel	Guinea pig	Galago
Mouse	-	1	0	0
Squirrel	1	-	?	0
Guinea pig	0	?	-	1
Galago	0	0	1	-

Analysis of Overlapping Inversions

We examined the micro-inversions found in ferret, dog, cat, and leopard species using pairwise dot-plots, and found that unusual features in these dot-plots are caused by a rare event of overlapping inversions.

We generated dot-plots of the regions surrounding each inversion locus in order to have a clear view of the local genomic architecture of the overlapping inversions. In order to remove confusion from repeats, we mask repeats from the dot-plots. Figure 2.24 shows the cat/dog, and cat/ferret dot-plots. Each dot-plot shows a clear inversion between cat/dog and cat/ferret. If this inversion happened in the dog, ferret ancestor, the dog and ferret sequences should be colinear in this region. However, the superposition of the two alignments in Figure 2.25 Left, shows that the inversions in dog and ferret correspond to different (overlapping) boundaries in cat, and the dog/ferret dot-plot instead shows multiple rearrangements. Micro-synteny blocks are outlined in red, green, cyan, and magenta in Figure 2.25Right. If we label the block order in dog 1 2 -3 4, the order in ferret is 1 -3 -2 4, requiring two inversions: 1 2 <u>-3</u> 4, 1 <u>2 3</u> 4 to transform dog into ferret. This also corresponds to the block order in cat: 1 2 3 4, as it takes one inversion to transform each of dog and ferret into cat, in accordance with the single inversion architecture shown in the dot plots in Figure 2.24. With these inversions removed from consideration our phylogeny does not separate ferret from the other species in this clade.

This presents a problem when determining the orientation of other species during the generation of a phylogenetic character. In this case it is arbitrary for species to be considered in the same orientation as dog (block 2), or opposite (block 3). Because of this, it is possible to create a bipartite inversion graph that has dog, cat, and leopard on one partition, and all other species on the other, thus creating an ambiguous character. Fortunately, we can detect such overlapping rearrangements prior to reconstruction. Currently we rely upon a sufficient quantity of valid characters so that erroneous characters may be removed using maximum conflict removal [8]. Further work is needed to incorporate the ability to model multiple genome rearrangements into our character generation and phylogenetic reconstruction.

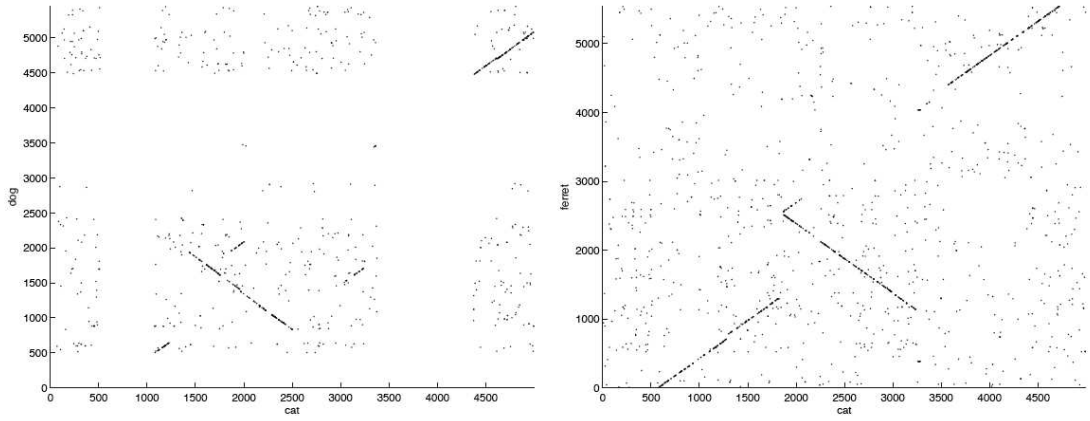


Figure 2.24 Dot-plots of cat versus dog (left), and cat versus ferret (right). Each dot-plot show a clear inversion against the cat sequence.

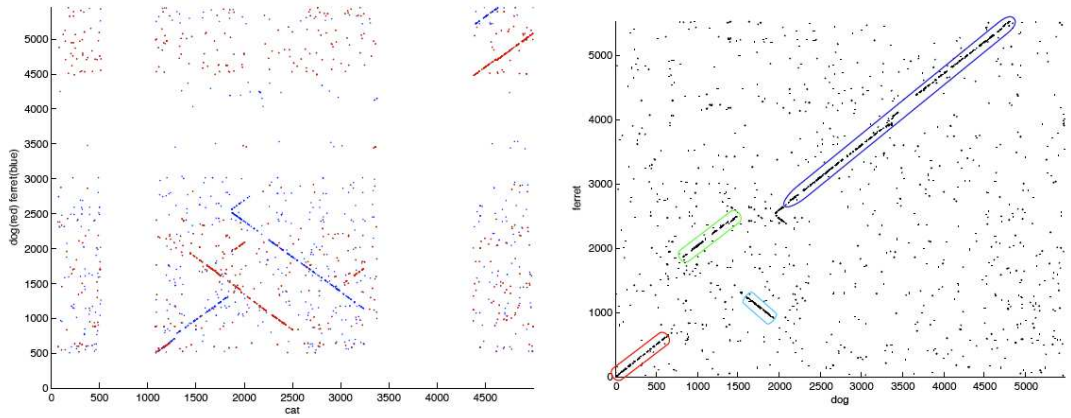


Figure 2.25 *(Left)* Superposition of the cat/dog (red) and cat/ferret (blue) dot-plots. Different regions in cat are inverted in dog and ferret, indicating that the inversion has taken place individually in the dog and ferret lineages. *(Right)* A dog/ferret dot-plot showing multiple micro-rearrangements between the dog and ferret genomes.

3

Part II Introduction: Fragment Assembly

3.A A history of fragment assembly

It is now common to think of a genome as a string of nucleotides, and present it as a string of the letters A, C, T, and G. With this abstraction it is easy to forget it is a single, long polymeric molecule. Sequencing a genome amounts to the determination of the chemical structure of a giga Dalton molecule, an impossible analytical task. Currently, the longest molecule that one may sequence is roughly a thousand bases. This means that it is impossible to read the sequence of a genome from start to finish in one piece, as one would read a book. Instead, the approach, pioneered by Frederick Sanger to sequence the bacteriophage λ genome [123], is to sequence many short random fragments of the genome, and computationally piece them back together, a method called *fragment assembly*. This is analogous to taking a book (rather many copies of the book), passing it through a shredder, and then reading the shredded fragments and taping them back in the original order of the book. As techniques for sequencing DNA have progressed, assembly methods have been tuned to analyze the sequencing data, and when the computational task was impossible, modifications were suggested of sequencing protocols to be more amenable to assembly. The research in fragment assembly may be seen as one of the greatest examples of cooperation between the computational and biological sciences

since researchers first began analyzing biological data with computers.

Methods for fragment assembly have evolved in step with the development of sequencing data since DNA sequencing was started by Frederick Sanger in 1975 [122, 124]. The Sanger sequencing method, for which all sequencing has been based upon until recently, measures sequences by replicating four samples of template in the presence of both nucleotides and dideoxynucleotides (ddNTPs) that cause early termination of replication upon incorporation. The sequence is then read by separating the four samples on a polyacrylmide gel. The first reads produced by Sanger were about 200 bases, and were from gel banding patterns deduced by eye [124]. The first methods written to assemble small genomes (under 6000 bases) by Roger Staden [136] to work with data collected from interpretation of gel images and manual input into data bases. A great technological advance in sequencing was achieved when radiolabeled ddNTPs were replaced by fluorescently labeled nucleotides and separation was done in using capillary electrophoresis [135]. The sequencing signal could then be processed by a computer to produce longer 400-base, and more accurate reads than sequencing on a gel. At the same time, Applied Biosystems began to produce industrial-grade sequencers, to rapidly expand sequencing capacity (`marketing.appliedbiosystems.com`). The methods for assembly also became more sophisticated. Around this time, the assembler SEQAID was written that formed the basis for most assemblers currently used [99]. Seeing the need to resolve more complicated sequences, in 1991, Al Edwards and Thomas Caskey proposed the use of *read-pairs* for assembly: pairs of reads sequenced from the same clone so that the approximate distance between the reads is known. Finally, in 1998 computational methods were introduced by Brent Ewing to add a quality score for each base in a read that reflected the probability that the base in the read is correct [38].

This part of this thesis is focused on methods to reconstruct genomes from reads, specifically reads that are considerably shorter than what past methods for assembly considered. The methods to be presented follow a long history of computational fragment assembly, beginning in 1977 with the methods of Roger Staden [136] to assemble very small genomes (under 6000 bases), with sequencing information entered by hand. Since then, the methods for assembly have become considerable more sophisticated. A time line of the development of assembly methods, as well as assembly milestones is

shown in Figure 3.1. Methods were steadily released during the early 2000's as whole-genome sequencing projects were nearing completion. The *de novo* assembly competition exploded in 2007 when the first High-Throughput Short Reads were released by 454 Life Sciences, and shortly followed by reads produced by Solexa/Illumina.

In 2001, two projects were completed that were landmarks for both Sanger sequencing, and fragment assembly: the separate publications of the draft sequence of the human genome [27, 144]. Separate, competing assembly routines were created for each project [69, 144]. The reads were produced over the span of several years, at the cost of hundreds of millions of dollars. The development of instruments for Sanger sequencing was already backed by strong research and development investment by industry with only incremental returns, and so researchers began to investigate ways to increase sequencing throughput with alternative analytical techniques to the Sanger method. Many methods were proposed such as Pyrosequencing [120], Sequencing by Polonies [133], and Sequencing by mass spectrometry [14]. The common theme to nearly all methods proposed as alternatives to Sanger sequencing is that a much higher sequencing throughput may be achieved at the expense of a shorter read length. Despite the shorter read length, and a slightly higher error rate [79, 51], the reads are fundamentally the same.

The sophistication of existing assembly methods made it prudent to ask: if the reads produced by alternative methods to the Sanger method are the same except shorter, is it necessary to develop a new method to assemble them? In effect, yes, as no single assembler has been universally applicable to all sequencing projects. For example, the competing assemblers: the GigAssembler [69], and the Celera Assembler [144] were both used to assemble the human genome, but considered different types of data. The GigAssembler performed hierarchical assembly first on small pieces of the genome sequenced in clones and ordered clones using a myriad of BAC paired-read data, mRNA transcripts, and radiation hybrid data. The Celera assembler performed whole genome shotgun assembly that lacked the hierarchical clustering of reads. The two types of data are illustrated/contrasted in Figure 3.2. Examples of other methods, are the Atlas assembler [52] which assembles a mix of hierarchical and shotgun based data, and the ARACHNE assembler [11, 61], which is designed to assemble mate-paired reads from libraries of several different clone sizes.

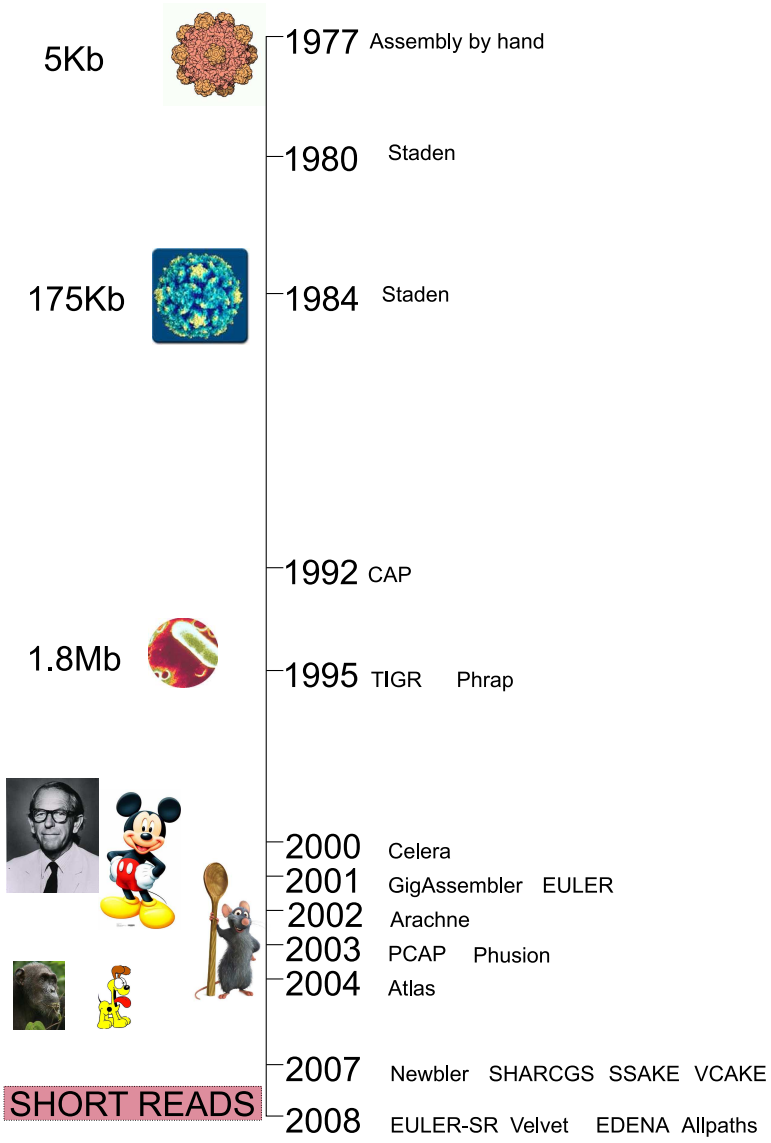


Figure 3.1 A time line of the development of fragment assembly software. The assemble of milestone genomes is shown to the left of the time line, and the release of assembly methods to the right. The order of genomes is: ϕX174, bacteriophage, *Haemophilus influenzae*, Human, mouse,rat, chimpanzee, and finally dog. There was a major increase in the rate that assembly programs were released when the use of high-throughput became widespread.

3.B Computational motivation of fragment assembly

To explain why new methods are required to assemble short reads, it is necessary to describe the computational formulation for which fragment assembly is based, and the approach many methods use to solve it. The framework is a type of problem that is NP-Complete, and cannot be efficiently exactly solved. The many methods that exist incorporate different types of data added to simplify the assembly problem, and are often particular to an individual sequencing project. For example, the *Rattus norvegicus* sequencing project used a combination of whole-genome-shotgun reads as well as sequenced BAC clones to produce a whole genome assembly.

The reason why so many assembly methods exist lies in the limitations of the solutions to the computational formulation for assembling genomes from reads randomly sampled with the shotgun method. Under the condition that reads are all sequenced without errors, the sequence of the genome must contain the sequences of all reads as substrings. One possible assembly is simply the concatenation of all sampled reads, but the correct assembly is assumed to follow maximum parsimony: the correct reconstruction of the genome is the shortest possible genome that contains all reads. Further restraints may be placed on this, such as appropriate gaps must exist between the placement of reads from opposite ends from the same clone. The problem of constructing a minimal length string containing set of substrings is the well studied problem, the *Shortest Common Superstring* (SCS) Problem. In the SCS problem, we are given as set of strings $\mathcal{S} = s_1, \ldots, s_n$ over an alphabet Σ, and one must find the shortest string s^* that contains each string in $\mathcal{S}$ as a substring. Unfortunately, for $|\Sigma| \geq 2$, the SCS problem is NP-hard [45]. There is no known efficient solution, and so any approach must be a heuristic. Furthermore, this abstraction hides the fact that there are nearly always errors in reads so that the genome does not need to contain exact matches to all reads, and reads may be sampled from either strand in the genome.

Approximation methods have been long studied for the SCS problem, of which many mention from fragment assembly of genomes as an application [142, 74]. The approach for fragment assembly that is motivated by the SCS problem was first described

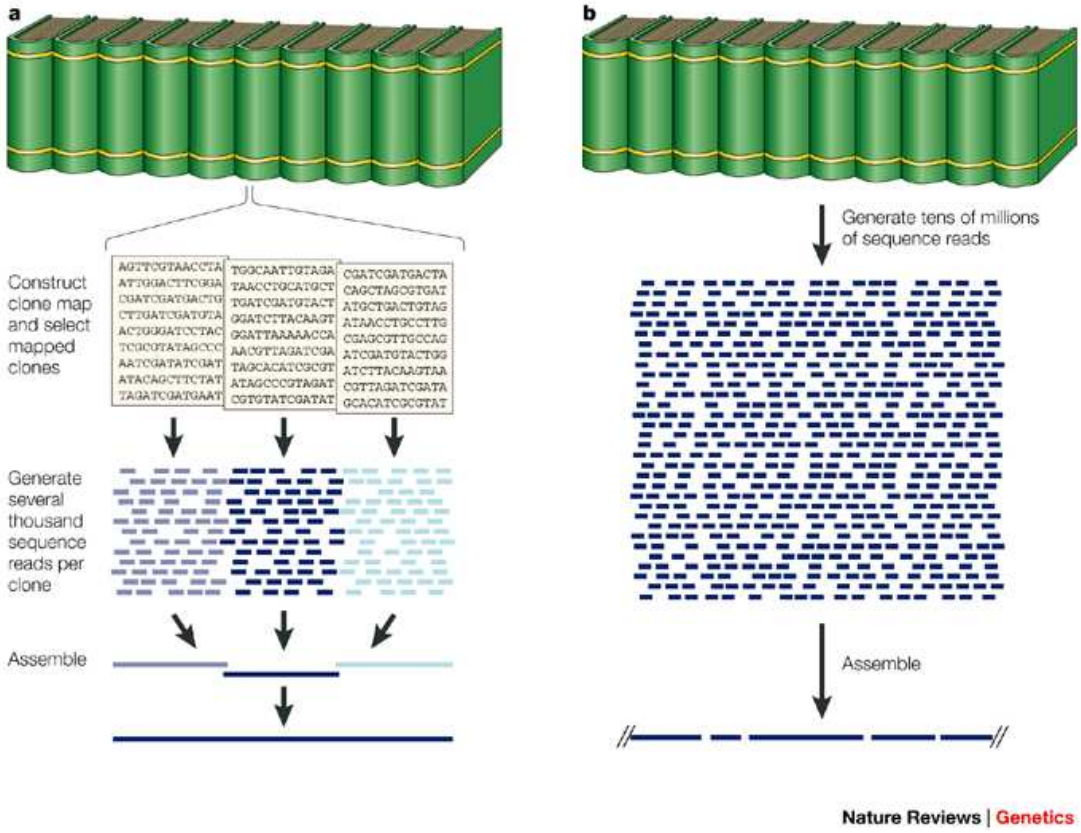


Figure 3.2 A diagram of whole genome sequencing strategies contrasting BAC by BAC (left) versus Whole Genome Shotgun (right). The BAC by BAC isolates BAC sized sequences from the genome, 40-400Kb, and assembles each sequence independently. The whole genome shotgun approach samples reads from the genome at random. In each approach, a sequence is assembled from reads sampled at random from the genome. The complexity of the assembly is reduced in the BAC-by-BAC approach, at the expense of the construction of a BAC library. Reprinted by permission from Macmillan Publishers Ltd: Nature Reviews Genetics [48], copyright 2001.

by Roger Staden [33], and later adopted by nearly all subsequent assembly methods including Phrap(`www.phrap.org`), the Celera assembler [92], ARACHNE [11, 61], ATLAS [52], and PCAP [56, 57], is the "overlap-layout-consensus" approach. This approach builds an assemblies in three phases: (*overlap:*) An *overlap graph* is constructed representing all pairwise overlaps of reads. An overlap graph encodes reads as vertices in the graph, and edges (u, v) are induced by overlaps between reads. Each edge is assigned a weight $w_{u,v}$ according to the length of the overlap. (*layout*) The layout of reads in the assembled genome is computed using the order that vertices are traversed in a maximum weight Hamiltonian path through the overlap graph. (*consensus*) The final sequence is determined using a consensus of overlapping reads.

To assemble the human genome, with over 3 billion bases and sequenced with millions of reads, a method must be efficient (near linear in time). There are efficient ways to compute all pairwise overlaps between reads [67] in $O(\epsilon nm)$ time, where ϵ is the expected error rate in reads, and so the overlaps, at least for Sanger reads, may be computed somewhat quickly. The next step, however, demands finding a Hamiltonian tour: a path through the graph that visits every vertex once. This is an NP-Complete problem, implying that an exact solution for even just a small bacterial genome will take more years to compute than there are particles in the universe! Paradoxically, there are many sequenced and assembled genomes orders of magnitude larger and more complex.

3.C Definitions of terms

Coverage the number of times a nucleotide is expected to be sampled in a read. This is $\frac{Nl}{G}$, where N is the number of reads, l is the length of a read, and G is the genome length.

Mate-pair a pair of reads sequenced from opposite ends of a clone. The distance between the two reads is approximately known.

3.D Approaches to fragment assembly

Under certain conditions, a greedy heuristic to determining the layout of reads is sufficient. For the sake of simplicity, assume reads are sequenced without errors, so that

the maximal overlap $ovp(a, b)$ between reads a, and b, is the longest length string s that is both a suffix of a and a prefix of b. When reads are sufficiently long, and sampled which high enough coverage that any two reads a, b that overlap in the correct layout overlap by at least p nucleotides, and the maximal length of a repeated sequence in the genome is r nucleotides, $p > r$, the following greedy heuristic will determine the correct sequence of the genome: choose a pair of reads (a, b) with the longest length $ovp(a, b)$, merge them into a new read $a' = \textit{prefix}(a) * ovp(a, b) * \textit{suffix}(b)$, where $a = \textit{prefix}(a) * ovp(a, b)$, and $b = \textit{suffix}(b) * ovp(a, b)$, and repeat until there is only one read. The layout of reads in the genome is determined by the order they are merged. To prove this method works, consider read a that has the longest overlap $ovp(a, b)$, but instead is merged with read c, so $a' = \textit{prefix}(a) * ovp(a, c) * \textit{suffix}(c)$. The sequence $ovp(a, b)$ must then occur twice in the assembled genome because a is not merged with b, and $|ovp(a, c)| < |ovp(a, b)|$. However since definition of the input constrains the maximal length repeat r to less than the overlap between any two adjacent reads, so any merging other than a and b is invalid. In the context of an overlap graph, the greedy heuristic will work when the maximum spanning tree is a nonbranching path from the vertex for a read at the start of the genome to the vertex representing the read from the end of the genome.

This greedy heuristic works on simple, nonrepetitive sequences, however the chances it will produce the correct assembly quickly deteriorate when there are repeats in the genome. As a simple illustration, a read ending in a repeat r, $a * r$ may overlap reads $r * b$, or $r * c$, to assemble into the sequences $a * r * b$, or $a * r * c$ and it may not be apparent which overlap to use without finding a Hamiltonian path. An example of this is shown on the left in Figure 3.3, where the greedy heuristic of choosing the longest overlap at each iteration causes the excision of a sequence (red-green) from the assembly. Furthermore, it may be possible that there are multiple, degenerate solutions (corresponding to multiple paths). An example of this is shown in Figure 3.3, where three copies of a repeat, and equal-weighted alignments of reads allow two possible Hamiltonian traversals of the overlap graph for two different layouts of the genome.

The approach most assemblers use to resolve this is to attempt to assemble

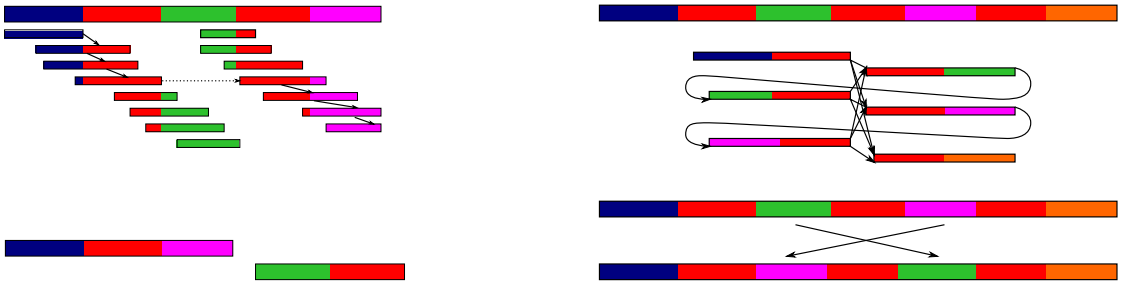


Figure 3.3 Two diagrammatic examples of repeats causing misassemblies in overlap graphs. (*left*) A repeat (red) causes the greedy approach to skip a repeat using a misleading maximal overlap (dashed line). (*right*) A genome with three copies of a repeat that has degenerate solutions.

all non-repetitive regions by masking known repeats [92], sequences that align to an unexpectedly high number of other reads [139], or sequences with l-tuples that appear more times than expected in the set of all reads, and then assembling all non-masked sequences [61]. This restricts assembly to building repeat-free sequences where the greedy heuristic is more likely to work. Each sequence that is assembled is referred to as a contig [99], since the genome is assembled into many small components.

There are several problems with this approach. First, the contigs must be ordered and oriented to produce the final assembly, and the repetitive sequence in between must be determined. This is accomplished in a *scaffolding* procedure, where a layout of contigs is determined that is concordant with mate-paired read information. This often results in a layout with misplaced (transposed), or misoriented (inverted) segments [60]. Another common problem with assemblies is that duplicated sequences with low divergence are frequently collapsed into one sequence [5].

Rather than masking repeats, it is preferable to formulate assembly in a manner that is able to preserve repeats. Such an approach is less likely to collapse repeats into a single contig, because the repeat may be resolved considering all read and mate-pair information, rather than early in assembly when finding overlaps.

3.E Sequencing by Hybridization

A different type of "read" was considered nearly 20 years ago in an application called *Sequencing by Hybridization* (SBH) [36]. In this method, an array of probes for all

combinations of DNA of length k is created. Fluorescently labeled DNA from a genome is hybridized to this array, allowing one to test for the presence of all words of length k (k-*mer*) in the genome by measuring probes that fluoresce. The set probes measured is called a *Spectrum*(k). The fragment assembly problem is then to construct the genome given *Spectrum*(k). Under the ideal conditions, *Spectrum*(k) is complete (no values are missing), and correct (only contains k-mers from the genome), and so only overlaps of $k-1$ need to be considered. The method to reconstruct the genome may again be formulated as the shortest common superstring, and solved by finding a Hamiltonian path through an overlap graph constructed with all vertices as k-mers, and all edges as exact overlaps of length $k-1$.

However, due to the particular structure of this problem: the overlap must be $k-1$, there is an alternative formulation for this problem based on an Eulerian path through a de Bruijn graph, first proposed by Pavel Pevzner [100]. Rather than representing overlaps as edges, they may be presented as vertices. The de Bruijn graph $G = (V, E)$ on a spectrum $\mathcal{S}$ contains a vertex for every $k-1$ mer contained in $\mathcal{S}$, and an edge (u, v) for ever k-mer in $\mathcal{S}$ that contains the $k-1$-mer u as a prefix, and $k-1$-mer v as a suffix. An Eulerian path through G, a path that visits every edge exactly once, will correspond to a reconstruction of the genome using SBH. An Eulerian path may be found in linear time, providing a practical approach to fragment assembly using Sequencing by Hybridization.

There is a drawback to fragment assembly using Sequencing by Hybridization: any repeat longer than $k-1$ bases in the genome will creates directed cycles in the de Bruijn graph that allow multiple Eulerian paths through the graph, each creating a different assembly of the genome. With the (overly optimistic) assumption that genomic sequences are generated as random sequences with each position having equal probability to be A,C,T, or G, the maximal length sequence that may be reconstructed with a spectrum of k-mers is $\sqrt{2 \cdot 4^k}$ [102]. Current technologies can produce arrays for $k = 10$, allowing reconstructions of 1.5K base sequences. Furthermore, studies of word frequencies in the human genome indicate that 40-base probes would be required to assemble much of the human genome [148]. With current technology, such a "microarray" would

require 1/5 of the surface area of the Earth![1]

3.F Eulerian assembly

A method to increase the effective tuple size was proposed by Ramana Idury and Michael Waterman [59] by applying the Eulerian path formulation for SBH to reads produced by the Sanger method. The spectrum $\mathcal{S}$ was generated using all k-mers from reads rather than an array. This limits k to the length of read overlaps rather than the length of probes used on DNA arrays. With 10× coverage with 600 base reads, the expected overlap is 540 bases, making the Eulerian approach appealing for fragment assembly.

As with other methods, there are some drawbacks to the Eulerian method for fragment assembly. Genomes are not generated by uniform independent random processes, and contain many repeats. The expected length of a genome that may be resolved by SBH (and therefore Eulerian assembly) is based off of an independent and identical distribution of nucleotides, so it overestimates real sequences. More importantly, each repeat creates a cycle in the de Bruijn graph, and the number of possible reconstructions of a genome grows exponentially in the number of cycles in the graph. An arbitrary Eulerian traversal of the de Bruijn graph is effectively guaranteed to give an incorrect reconstruction of the genome. Rather than finding an Eulerian traversal of a de Bruijn graph, the graph is *condensed*: paths along non branching vertices are replaced by a single edge labeled by the sequence that generated the path, and the assembly is reported as the set of edge labels. For example, a path: $(GACT, ACTT), (ACTT, CTTG), (CTTG, TTAA), (TTAA, TAAG)$ may be condensed to an edge $(GACT, GCGA)$ labeled by $GACTTGAG$. A very similar step is done by overlap-graph based assembly methods to remove transitive edges in the overlap graph up to repeat boundaries, and to find the consensus of sequences along non-branching paths [91].

Second, because the de Bruijn graphs are constructed from a random sample of reads that represent random, overlapping substrings from the genome, multi-edges

[1] Assuming an optimistic estimate of 1 M probes per/cm^2.

(edges with the same start and end vertices) are replaced by a single edge. A de Bruijn graph (without multi-edges) that has cycles will not have an Eulerian traversal: a path that visits every edge *exactly* once. To transform the de Bruijn graph into a graph that has an Eulerian path, edges corresponding to repeats in the genome must be replaced by multi edges with a count equal to the multiplicity of the repeat in the genome. The multiplicities may be determined as a solution to the Chinese Postman Problem [44]: a path in a graph of minimal length that visits every edge *at least* once.

Simply outputting the edge labels of a condensed de Bruijn graph will typically produce a very fragmented assembly. This is because there is information contained in reads and mate-paired reads about the correct Eulerian path in the de Bruijn graph that is lost when only adjacent k-mers from reads are considered. Because the de Bruijn graph is constructed from overlapping k-mers in reads, each read corresponds to a unique path in the graph, called a *read path*. The reformulation of fragment assembly as an (arbitrary) Eulerian path in a de Bruijn graph from a set of reads (of length $> k+1$) is incomplete. In the ideal (and very artificial) case when reads do not have errors and the multiplicity of every multi edge is known, the problem is to find an Eulerian path that contains all read-paths as subpaths. Under the assumption of parsimony the length of the path must be of minimal length. This problem was considered in [106, 104], as the *Eulerian Superpath Problem* (ESP), although the presumption of a minimal length path was not explicitly stated. This problem was formulated as the *De Bruijn Superwalk Problem* [82] with the explicit requirement that the superwalk be of minimal length, and was proved to be NP-hard.

The approaches taken to restore information from reads lost by creating the de Bruijn graph in [59], and similarly the solution to the Eulerian Superpath Problem [106, 104] are based on a set of graph transformations. Rather than solving for an Eulerian path, the graph is iteratively transformed into a more simple graph with fewer vertices and fewer, longer edges. The new edges must preserve the sequence information from all reads. The transformations defined in [59] are the *elimination of forks* and *elimination of crosses*. Examples of these are shown in Figure 3.4.

The transformations defined in [106, 104] are much more sophisticated. All transformations are done using detach operations 3.5(a $\rightarrow$ b). The idea behind the

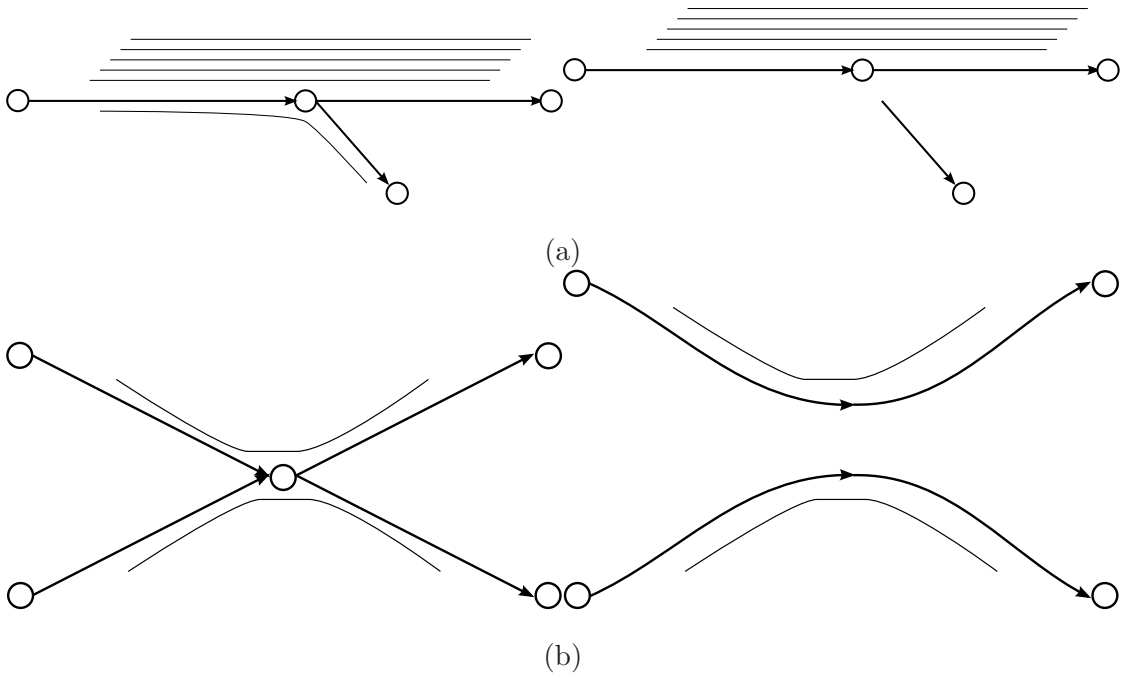


Figure 3.4 *(a)* Elimination of forks. When a branch has an unexpectedly low number of read paths that map to it, it is likely spurious and should be removed. *(b)* Elimination of crosses. Crosses happen from repeated k-mers. The elimination of a cross routes edges out from the repeated vertex, when path information is not conflicting (each exit edge is paired with only one entrance edge).

heuristic is that three serially connected edges a,r,c may be joined into a single edge containing the sequence of a, r, and c, if there is no path that contains any other edge d following a.

The final difficulty for Eulerian assembly is that errors in reads add extra edges to the de Bruijn graph that fragment the assembly. Fortunately, it is possible to detect and correct errors in reads prior to assembly, as shown in [106].

Assembly using de Bruijn graphs has a practical advantage over assembly on an overlap graph for high coverage: space efficiency with compact representation of overlaps. The graph may be constructed in O(N) time where N is the space taken by all reads, and in O(G) space, where G is the length of the genome. This is important because the number of overlaps scales with the number of reads sampled, and as a conservative estimate high-throughput short read sequencing projects sample an order of magnitude more than in traditional Sanger sequencing projects. To estimate the number of overlaps of any given read, let the sampling rate f be the number of reads per base in the genome that are read. If reads are sampled at a uniform distribution from the genome so that a read will start at a position x in a genome of length G with a probability $\frac{1}{G}$. There are $fG = N$ reads sampled from the genome, and since typically N is large (100,000,000 for an Illumina sequencing run), and $\frac{1}{G}$ is small, the expected number of reads starting at a position may be approximated by a Poisson distribution with mean f. The number of reads expected to begin in an interval of length p is pf. In high-throughput sequencing projects, a conservative estimate for f is 3. With short, $l = 35$-base reads, if the minimum allowed overlap length k is 20, the expected number of reads to overlap a given read is $(l - k) * f = (35 - 20) * 3 = 90$. By comparison, there are far fewer reads in a Sanger sequencing project: $(600 - 40) * (0.015) = 7.3$ (600 base reads, a 40 base detectable overlap, and 8X coverage). This is an increase in the number of overlaps *per read*. For the purpose of an efficient implementation, a sequencing project with 10^8 reads there will have $\sim 10^{10}$ overlaps, and assuming 8 bytes per overlap, a memory requirement of 80G bytes.

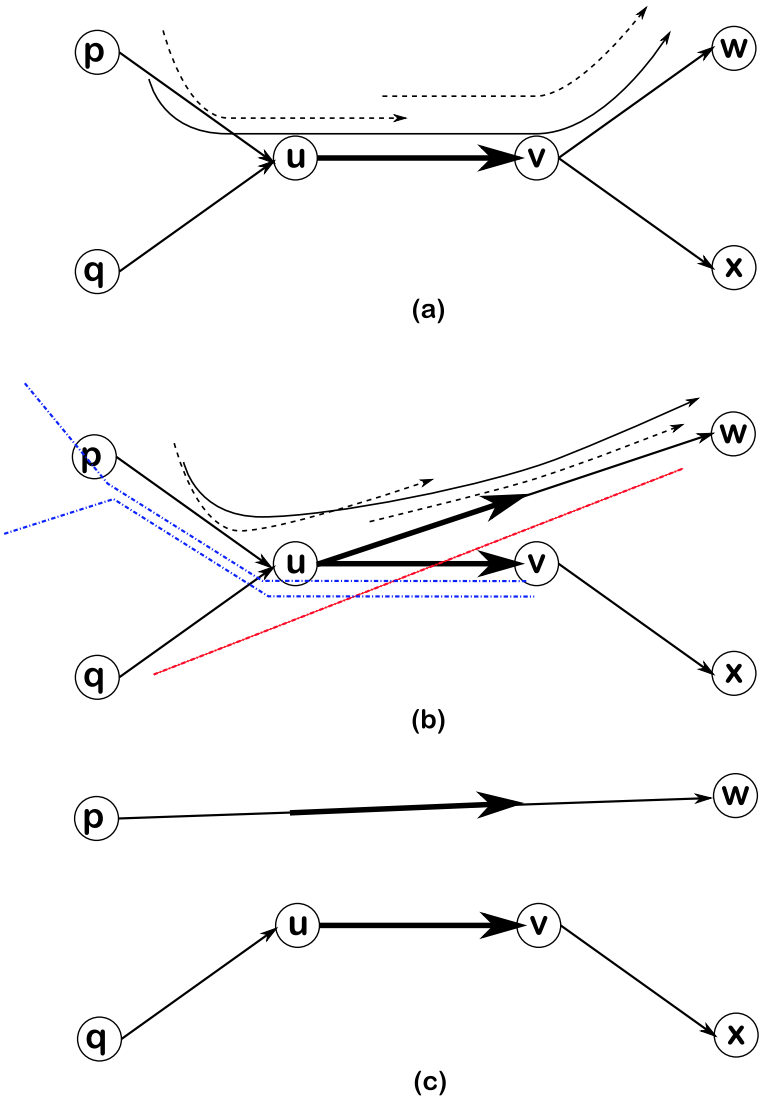


Figure 3.5 A diagram of the equivalent transformation operation defined in [106]. *(a)* All paths covering an edge are considered when performing transformations about that edge. *(b)* The effect of the detach operation, where an edge is detached from a vertex, and joined to a previous vertex possibly creating one fewer branch in the graph. All paths are updated to be consistent with the detached and joined edges. Alternative possible paths are shown that may be conflicting (red), or ambiguous (blue). *(c)* A detach operation may be performed only if there are no conflicting paths or ambiguous paths.

3.G Description of short read sequencers

454 Pyrosequencing Reads generated by the Pyrosequencing method. Current reads are up to 400 bases, however the original GS20 reads were 80-120 bases.

Illumina Genome Analyzer Sequencing-by-synthesis on templates amplified after being fixed on a chip. The first commercial reads were 35 bases, and by 2009 Illumina intends to release sequencing kits to produce 100 base reads.

Applied Biosystems SOLiD Sequencing by ligation based method. This method uses a special "colorspace" encoding of reads since 16 probes are colored with 4 different fluorescent labels. Each read may be decoded based on the first nucleotide pair in the read, however errors in the read change the interpretation of the entire read.

Helicos HeliScope Single molecule sequencing. At the time of publication of this thesis, there is no commercial availability of a Helicos sequencer, however they have published proof of concept [51].

3.H Outline of thesis

In Chapter 4, a pioneering study is presented that examined the feasibility of short read fragment assembly. This first proposed the return to de Bruijn graphs to assemble short reads. Furthermore, this work presents a dynamic programming method for error correction that reduces the errors even highly erroneous reads (4%) by three orders of magnitude.

In Chapter 5, the method EULER-SR is presented. This method is based off of a combination of the de Bruijn based assembly [106], and A-Bruijn based assembly [107], which uses graph-simplification methods to change the graph topology and correct errors in reads. The methods were modified to use a Maximal Arborescence to define the correct portions of the graph, in contrast to the Maximal Spanning Tree that is used in [107].

In Chapter 6, the extension of EULER-SR to assemble using mate-pairs is described. This uses an approach similar to [104], except the method to transform mate pairs: "read1 - gap of length d - read2" to mate-reads "read1 - sequence of length d - read2" is modified to take advantage of high coverage and fill in the sequence between

reads even in complicated repeats. Also, this paper introduces a technique called "read threading" that is able to correct highly erroneous reads, as long as they have a high quality short prefix.

Finally, in Chapter 7, several future directions for research in *de novo* fragment assembly are presented.

4

Fragment assembly with short reads

In this chapter, we present a preliminary study conducted on 2004 that examined the feasibility of fragment assembly for short reads. At the time, several papers had been published about methods for producing short reads, but no full scale studies had been conducted. Although there were many compteting technologies, the most likely first "Next Generation" non-Sanger sequencing technique was to be based on Pyrosequencing, developed by Mostafa Ronaghi while at Stanford. It was known this method would produce reads much shorter than those of the Sanger method, and that there would likely be many indels, however little else was known. We proposed that since Eulerian assembly was based off of Sequencing by Hybridization, which is effectively assembling short, 8-12 base reads, it was well suited for assembly of short reads. We tested this by simulating data from human BACs, and assembling them with a modified version of the EULER assembler [106]. We found it was necessary to produce a new error correction routines for correcting reads that have a large number of indels.

4.A Introduction

The Sanger sequencing method has enjoyed great success since its debut in 1977, as it is used in virtually all sequencing projects. The amount of sequence information has exponentiated due to advances in automated sequencing technology and investment

from the Human Genome Project. State of the art automated sequencers can sequence a staggering 3 mb/day. However, despite its strong points, there are some drawbacks to Sanger sequencing. First, it is dependent on clone libraries for sample preparation, and some genomes have regions not amenable to cloning. Also, future genomics projects may require sequence throughput orders of magnitude faster than what is currently possible. Because of this researchers are searching for alternative sequencing techniques that would bypass the shortcomings of Sanger sequencing ([120], [14], [36], [112]). These techniques aim at *clone free* sequencing to enable sequencing of regions that are not amenable to cloning, and to dramatically decrease the time and cost of sequencing. Almost all the new techniques produce short reads, i.e. 80-200 bases, as opposed to the 500-750 length reads produced by Sanger sequencing. It is hardly believed that such short reads can be assembled efficiently; and efforts are being made on increasing the read length. In this chapter we examine the limits of assembling short reads, and estimate the amount of additional finishing efforts that will be be required for assembly.

The assembly of short reads is directly applicable to several new non electrophoretic sequencing technologies being developed, such as Pyrosequencing^{TM}[120, 118] mass spectrometry based sequencing [14], seqencing by Polonies [84], and single-molecule sequencing [17]. These have the goal of sequencing genomes several orders of magnitude faster than what is possible with electrophoretic methods. The feasibility of using non-electrophoretic methods to sequence a long DNA molecule was recently demonstrated by the *454 Life Sciences Corporation* by the sequencing of Adenovirus. This resulted in the first DNA sequence entry in GenBank that was sequenced by a technique alternative to Sanger sequencing [131]. The reads produced by new sequencing techniques differ in both read lengths and quality profiles characteristic of electrophoretic sequencing. In particular the read length is sacrificed for much higher throughput, resulting in high coverage with short reads.

The success of the new high throughput whole genome shotgun sequencing technologies will depend not only on the speed of the analytical process of producing reads, but also the solution of the computational problem of assembling sequence fragments into complete genomes. The common approach in many fragment assembly programs, such as Phrap [38], CAP [56], PCAP [57], and the TIGR assembler [139], is the overlap-layout-

consensus approach. The new genome assembler from Celera [92], and the ARACHNE assembler ([11], [61]), mask the repeat regions in the reads in order to reduce the complexity of the overlap graph and alleviate the difficulty of layout stage. This idea is successfully applied to assembling large eukaryotic genomes with whole genome shotgun reads, including fruitfly [1], human [144] and mouse [28]. The major difficulty in assembly is the presence of repeats that are longer than the typical read length. Masking out repeats in such an approach leads to a fragmentation of the overlap graph. Therefore the shorter the read length the larger the number of repeats that present difficulties to assembly, and a greater enhancement of loss of information of the repeat masking procedure in ARACHNE and Celera assemblers. To demonstrate the increase of the assembly complexity with the decrease of the read length, we computed the number of perfect overlaps found in hypothetical sequencing projects with varying read lengths, shown in Figure 4.A. While for a given genome the number overlapping reads scaled roughly linearly with read length, the number of repeat pairs increased dramatically in the larger and more complex *Neisseria meningitidis* (NM) genome. With short reads, assembling a BAC became as complicated as assembling a bacterial genome with normal reads, even when reads have no sequencing errors. The assembly of the *Neisseria meningitidis* (NM) genome, known to have a large number of repeats, became a formidable problem. In practice, it was difficult to assemble even very short genomes using short reads using Phrap[1], an excellent program to assemble BACs with normal read lengths. Therefore scaling short read assembly to longer and longer genomes presents a difficult computational problem. The recent announcement of Adenovirus sequence by *454 Life Sciences Corporation* did not reveal the computational approach behind this advance. However, we remark that the Human Adenovirus E has only one repeated region 51 bases long, and can otherwise be modeled as a "random" sequence. This 51 bp long repeat does not present a problem to assembly because it can be bridged by even extremely short reads; Phrap was able to assemble this with no modifications to the code. Moreover, one would be able to assemble the equivalent of a human genome with short reads if the human sequence followed a random distribution. Therefore, even after the *454 Life Sciences*

[1]Phrap produced only a few very short (100 bp) contigs as well as a core-dump when assembling a BAC with short reads. The adenovirus genome was assembled using less than 20X coverage.

Corporation announcement of the Adenovirus genome, sequence assembly of short reads for more complex genomes (even BACs) remains an open problem.

In an extreme case of high enough coverage and short enough reads, the fragment assembly problem becomes effectively the same as sequencing by hybridization. In this extreme case the read length is fixed (e.g. $l = 20$), and there is one read starting at every position in the genome. The Eulerian Path approach to solve sequencing by hybridization was presented by [100], and proposed for the assembly of regular reads by [59]. This has been implemented in the EULER fragment assembly program [106] and further developed in [104]. Euler has the advantage over assemblers based on overlap-layout approach since it meticulously works with repeats, and instead of masking repeats, generates a repeat graph that reveals the underlying structure of the repeats within a genome. The repeat graph can direct biologists towards experiments that allow the resolution of repeats [111].

4.B Alternative sequencing technologies

4.B.1 Pyrosequencing

Pyrosequencing^{TM} [120, 118] is a non-electrophoretic DNA sequencing technique that uses sequencing-by-synthesis. In this method, a solution of an amplified sequence and corresponding primer is mixed with four enzymes: polymerase, ATP sulfurylase, luciferase, and apyrase. Sequencing is carried out by iteratively adding solutions of alternating nucleotides (A,C,T,G,A,C,T,G...). When a nucleotide is incorporated into the template strand by polymerase, PPi is released and subsequently converted into ATP by ATP sulfurylase. This drives a light emitting luciferin oxidation that can be detected by a light sensor. Unincorporated nucleotides are degraded by apyrase. The light intensities measured upon the addition of nucleotides are combined into a pyrogram from which the sequence can be read. Pyrosequencing has been used effectively to sequence short primer extensions for SNP detection [39]. It remains to be seen what the error rates are of Pyrosequencing, however it is feasible that it is higher than that of electrophoretic methods.

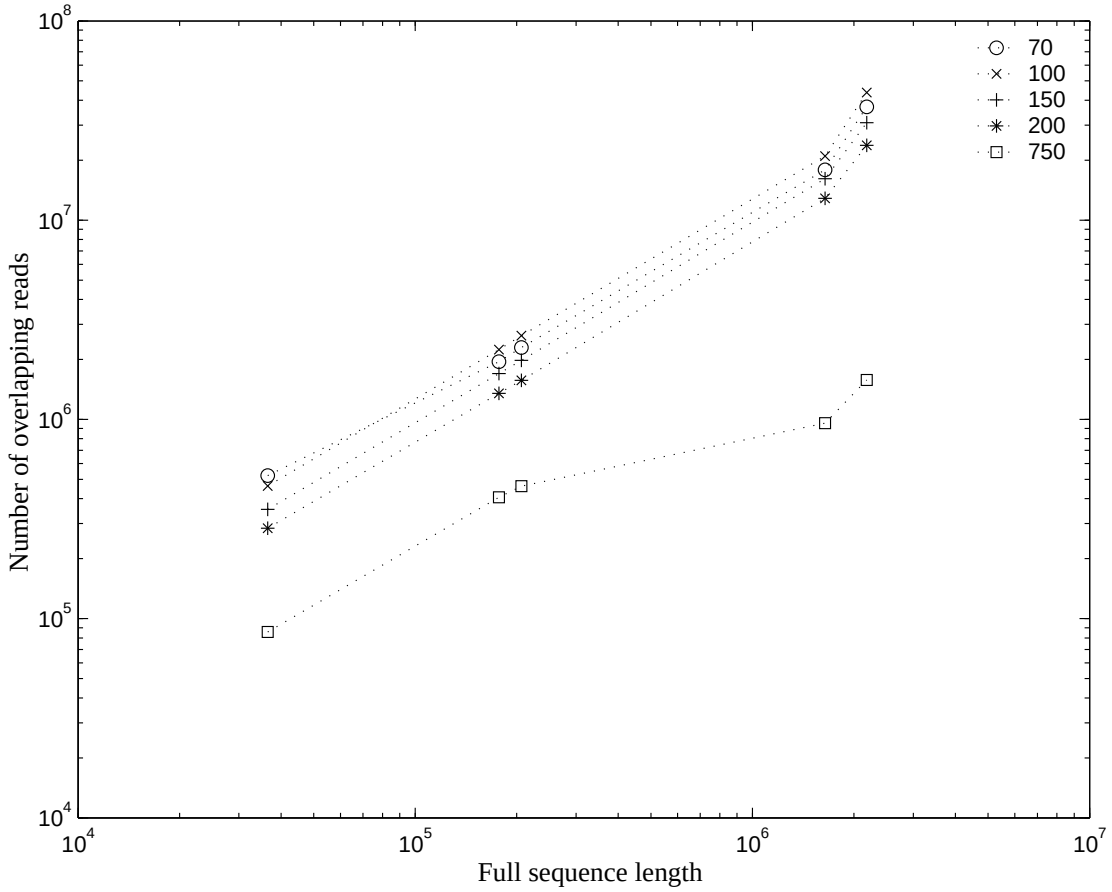


Figure 4.1 Number of overlaps versus complete sequence (genome) length. Data were sampled at 30X coverage from Adenovirus, *CJ*, *NM*, and two BACs. Overlaps were found using megablast [152] and filtered for exact overlap alignments at least 30 nucleotides long. Read length 750 was chosen as a reference of the assembly complexity when assembling reads with current read length.

4.B.2 Sequencing by MALDI-TOF

A recent development in *de novo* DNA sequencing by Matrix Assisted Laser Desorption-Iionization Time of Flight (MALDI-TOF) mass spectrometry has been described by [14]. In this technology sequences of up to 200 nucleotides are partially digested by base specific endonucleases, and the resulting fragments are analyzed through mass spectrometry. Each peak in the mass spectra corresponds to a number of nucleotides called compomers. By solving the Sequencing by Compomers problem proposed by Böcker [14], the spectra are resolved into high quality reads. The detection of compomers through MALDI-TOF has been used for SNP discovery [13], and is implemented in the SEQUENOM MassArray platform.

4.C Assembly of error-free reads

Errors in read data greatly complicate the task of fragment assembly. It is possible to correct many of the errors in reads prior to assembly. However as a first step in proof of concept for short read assembly it is necessary to show the upper bound on the resolution that can be achieved in assembly of short reads. A modified version of EULER was used to assemble the reads.

As described by ([106],[104]) the Eulerian approach to fragment assembly resolves a sequence by finding an Eulerian path through a de Bruijn graph representation of a genome. Let G_l (G_{l-1}) be the set of l (l-1) tuples present in a genome G. To form the de Bruijn graph a vertex v labeled $\langle t_1 \dots t_{l-1} \rangle$ is created for each $l-1$ tuple in G_{l-1}, and for each l-tuple $t \in G_l$ a directed edge ($\langle t_1 \dots t_{l-1} \rangle$, $\langle t_2 \dots t_l \rangle$) is added. The sequence corresponds to an Eulerian path through this de Bruijn graph. In a sequencing project G_l (and therefore G_{l-1}) is approximated by the set of l-tuples present in a set of reads. Repeats in a genome create tangles in the de Bruijn graph [104]. The equivalent transformation approach proposed by Pevzner et al. [104] uses reads and mate pair information if available to define paths through the read-generated de Bruijn graph, and thus resolve many tangles. Tangles that remain require additional finishing experiments to be resolved.

In this study we assembled simulated reads from several test viral genomes,

BACs, and bacterial genomes: Adenovirus (Adeno), BAC 85H09 , BAC 47A01, *Campylobacter jejuni* (CJ) , and *Neisseria meningitidis* (NM).[2] The genomes were sampled for 30× coverage, and ignoring end effects, all test cases had full coverage. The contigs resulting from assembly for various read lengths are listed in Table 4.1. The number of contigs drops at a decreasing rate with increasing read length. BAC 47A01 contains many low complexity regions that were difficult to assemble with the shortest read length. Also, the number of contigs may be artificially large since many of the contigs generated for short reads are short, and most of the genome is contained in a few long contigs. For example, the number of contigs that give 95% coverage using reads averaging 70 bases for BAC 85H09 is 9; the largest 10 contigs for BAC 47A01. In CJ, 21 contigs produced over 95% coverage when assembling reads averaging length of 70 while 11 contigs covered at least 95% of the genome when assembled with 750 base reads. The complex nature of the NM genome was reflected in the smaller size of contigs; 95% coverage required 344 contigs assembled from 70 bp reads (20% of all contigs generated), however only 28 contigs were required for this coverage when assembled from 750 base reads.

[2]NCBI accession: (Adeno) AF394196, (85H09) AC084390, (47A01) AC123854, (CJ) CJ11168X1 (NM) NMA1Z2491

Table: 4.1 Assemblies of sample genomes using read lengths averaging 70, 100, 150, 200, and 750, all ± 10 bases. Paths consisting entirely of nodes of degree 2 are collapsed into one edge, representing the theoretical limit of resolution using the specified read length. For each read length, the following are reported: {*rep*} the number of edges in the repeat graph (de Bruijn graph) constructed from the test genome using tuples equal to the average read length size + 10, {*asm*} the total number of contigs generated from assembling simulated reads, and {*≥1K(%)*} the number of contigs greater than 1kb with the percentage of the genome the contigs cover.

		70			100			150			200		
Genome [Length (kb)]	rep	asm	≥1K(%)	rep	asm	≥1K(%)	rep	asm	≥1K(%)	rep	asm	≥1K(%)	
Adeno (35)	1	1	1(100)	1	1	1(100)	1	1	1(100)	1	1	1(100)	
85H09 (177)	16	25	10(98.9)	10	20	7(98.9)	10	12	5(99.5)	7	12	5(99.6)	
47A01 (206)	69	59	13(98.2)	25	28	6(98.9)	29	13	5(99.0)	13	13	4(99.3)	
CJ (1,662)	73	72	37(99.5)	63	58	34(99.6)	54	48	29(99.5)	45	48	22(99.5)	
NM (2,184)	1803	1864	309(93.4)	1299	1247	299(94.6)	563	296	215(96.2)	266	296	114(97.8)	
		750											
Adeno (35)	1	1	1(100)										
85H09 (177)	1	1	1(100)										
47A01 (206)	1	1	1(100)										
CJ (1,662)	22	27	22(99.9)										
NM (2,184)	92	59	48(99.7)										

4.D Assembly of reads with errors

As mentioned in [104], the EULER assembly method works well on error-free reads, however sequencing errors in reads can complicate the de Bruijn graph making it difficult to determine the true sequence. This can be alleviated by detecting and fixing errors prior to sequencing.

Rather than using alignment and base-calling during a post-processing (consensus) step of fragment assembly, error correction is performed prior to assembly through solving the Error Correction Problem [106]. In this problem, let G_l be the set of l-tuples in a genome G. A read that has errors will contain l-tuples not in G_l, and is transformed so that it only contains l-tuples in G_l. The transformation used is the Spectral Alignment Problem (SAP) [106]. In the SAP, one is given a collection of l-tuples, T, and a string. A string is called a T-string if all its l-tuples belong to T. Let $\mathcal{T}$ be the collection of all T-strings. The SAP is to find $s^\star$ such that the edit distance $d(s, s^\star)$ is minimal for all $s^\star \in \mathcal{T}$. Error correction is performed by setting T to G_l, and applying spectral alignment to each read. In *de novo* sequencing G_l is not known, and an approximation is made by choosing a threshold M then labeling any l-tuple present more than M times throughout all reads as 'solid', and 'weak' otherwise. An example of the distribution of l-tuple multiplicities for sample reads from the CJ genome is given in Figure 4.D. For this genome, we can use a threshold $M = 10$ that will eliminate false positives, erroneous l-tuples that should not be in G_l, and minimize false negatives, l-tuples in G_l that do not have the multiplicity to be solid.

An approximation to the solution for the Spectral Alignment Problem is currently used in the EULER assembler. In the approximation, all unambiguous changes are made that can transform weak l-tuples into solid ones. This method performs well on the read lengths and error profiles of modern sequencing projects.

High coverage, repeats, and high error rates make it difficult to determine which l-tuples should belong in T. Fluctuations in coverage by shotgun sampling require a low threshold M so that valid regions are not considered weak. Unfortunately with a large number of reads and high error rates this will also include many false positives. Increasing l will decrease the false positive rate, but in reads with high error rates this

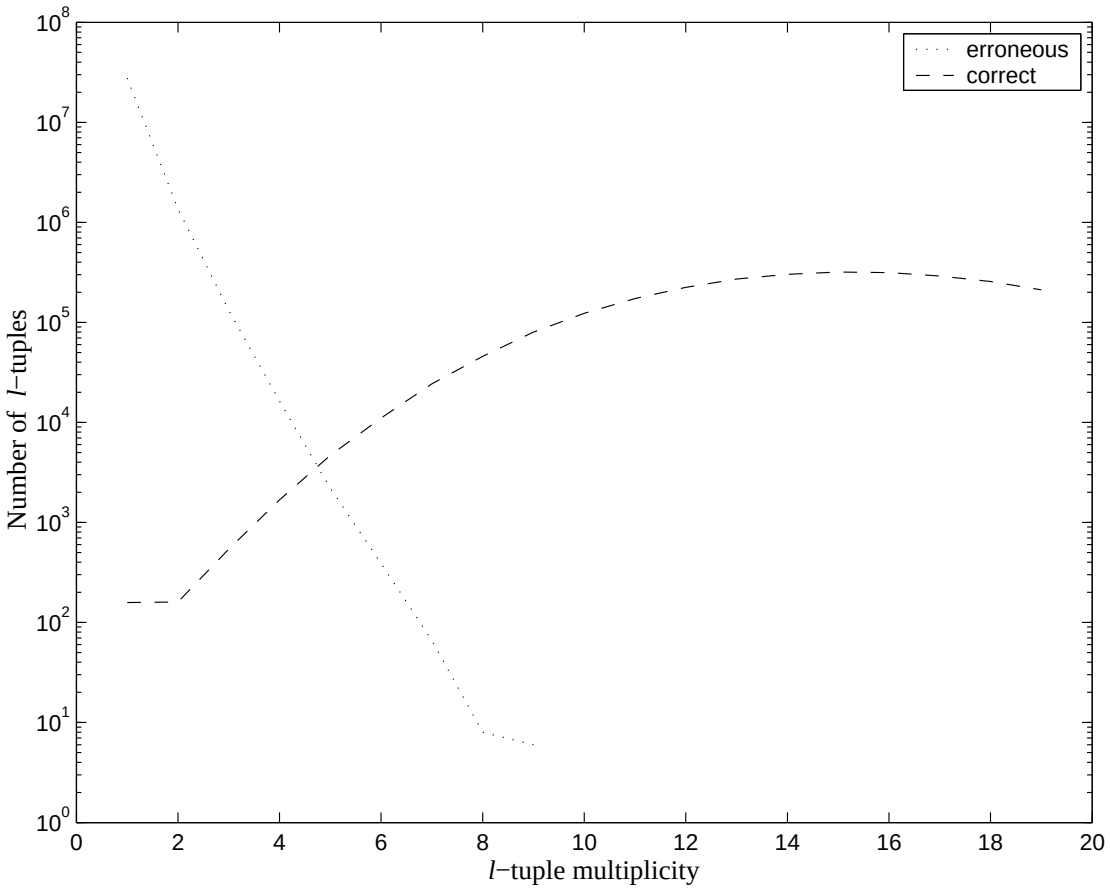


Figure 4.2 Number of erroneous and correct l-tuples found in simulated reads from the Campylobacter jejuni genome versus multiplicity. The genome was sampled at 30X coverage with average read length of 100. Reads were corrupted with a 2% error rate uniformly distributed between mutations and insertions/deletions. Most erroneous l-tuples are present in low multiplicities.

makes it difficult to determine which changes are appropriate to transform the read into a T string. A dynamic programming solution allows for choosing a small M while eliminating many false positives since it considers multiple possible modifications to a string before selecting the optimal ones.

4.E Dynamic programming solution to Spectral Alignment Problem

In the context of resequencing by hybridization, [98] studied a problem similar to SAP where a dynamic programming solution was presented, although it was not formulated as SAP. This approach finds the highest probability path through a de Bruijn graph constructed from probe matches in an universal array. While the approach taken in [98] is similiar to our approach, it cannot be directly applied here. First, the scoring scheme is probabilistic rather than discrete, although this may be easily updated to fit the edit distance score presented here. More importantly, the search for optimal path transitions from 4^l possibilities are considered at each iteration. This is computationally tractable for universal arrays, where l can range from 6 to 9, however in the case of error correction such small values of l can lead to saturation of the T-spectrum so that nearly all 4^l l-tuples are solid. In this approach values of l range from 15 to 20, and so it is important to only search through a subset of the 4^l possibilities. Here we describe a complete solution to SAP, and give a heuristic that solves the SAP in most cases without searching this complete search space.

In our solution we use the notation $\vec{a}$ as a right-shift of the l-tuple a by i characters, and $X* \vec{a}$ as the concatenation of the character X with the resulting $l-1$-tuple. For example, if a is the 4-tuple $ACGT$, and X is G, then $X* \vec{a}$ is $GACG$. For a string s and an l-tuple a from spectrum T, we define a score $score(i,a)$ to be the minimum edit distance fo the prefix of s of length i and all T-strings ending in a (if a is not in T, score(i, a) is infinity). This may be computed recursively using the following relation:

$$\text{score(i, a)} = \min_{\text{X} \in \{\text{A, C, T, G}\}} \begin{cases} \text{score(i} - 1, \text{X}* \vec{\text{a}}) + \begin{cases} 1 & \text{if} a_l \neq s_i \\ 0 & \text{otherwise} \end{cases} \\ \text{score(i} - 1, \text{a}) + 1 \\ \text{score(i, X}* \vec{\text{a}}) + 1 \\ L(s_i, a) \text{ if i} = \text{l} \end{cases}$$

.

The inital condition is $score(0,a) = 0$. Denote hte length of string s as $|s|$.

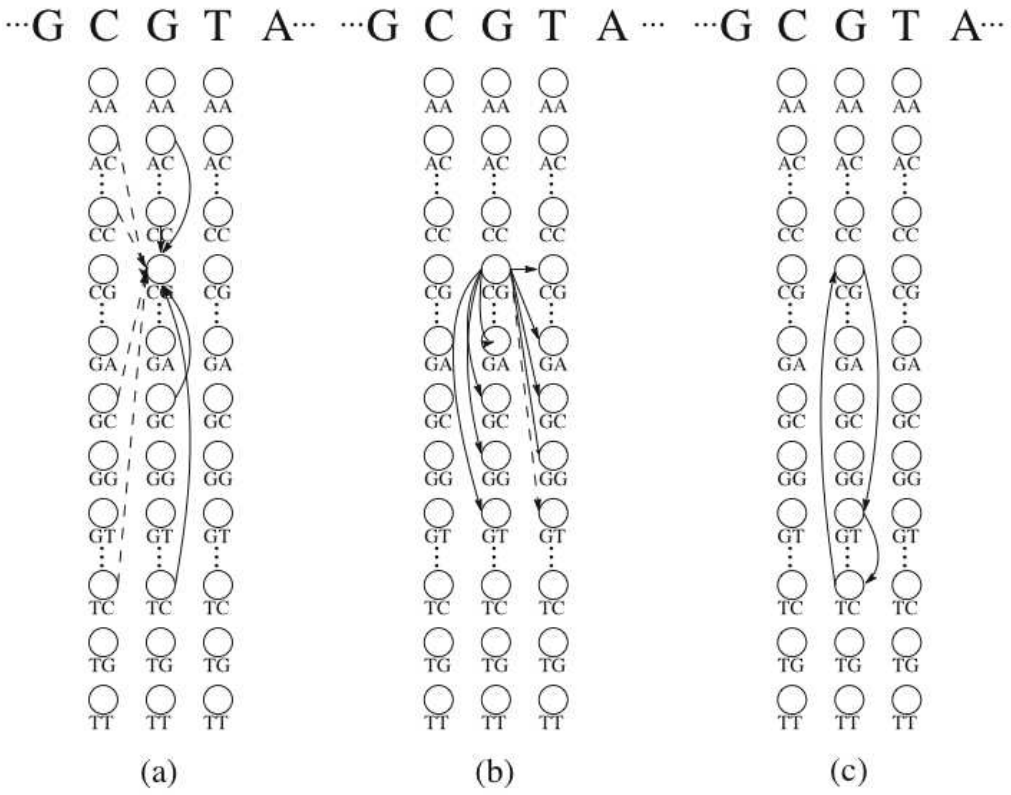


Figure 4.3 Example of a graphical solution to the SAP. Dashed edges have a score of 0 and solid edges 1. Left, vertices used to compute the score of vertex CG. Center, vertices that reference CG. Right, a positive cycle. The complete cycle will not be used in a shortest path.

The minimum $score(|s|, a)$ for all $a \in T$ corresponds to the minimum cost T-string for s. This recursion may be represented asa graph where each vertex cofresponds to a prefix of length i and an l-tuple a. Edges connect vertices referenced by a recurrence relation. The in-degree of a vertex is equal to the number of non-inifite score vertices referenced to compute its score (for $i = 0$ the score is 0), and the out-degree is equal to the number of times the vertex is used in a recurrence relation (0 for vertices corresponding to $s_{|s|-l}$. The minimum score T-string corresponds to a shortest path from any vertex with in-degree of 0 to any vertex with out degree of 0. All edges have non-negative weight, and so there are no negative cycles, and the shortest path may be computed using any shortest path algorithm such as Bellman-Ford [31]. An example of t he graph representation of the recurrence relation is shown in Figure 4.3.

We applied heuristic approximations to the SAP in order to improve performance in fixing errors in projects using short reads. When performing spectral alignment for reads with a known genome, the spectrum is set to G_l . Since G_l is not known in a *de novo* sequencing project, it is approximated by the set G_l^M , all l-tuples with multiplicity not less than M within all reads. Although the size of G_l^M is smaller than the total possible 4^l l-tuples, it is desirable to reduce the search space for T-strings even further. If a string s contains a substring u that is a T-string, we only consider T-strings containing u. The scoring function for this is $\text{score}^u(u, i, l)$, the minimum edit distance for string s to all T-strings containing u. Since this is easily computed if u is a prefix of s, we can solve the SAP for a substring of s such that u is a prefix of s. In our implementation we set u to be the first solid substring in s of length $l + \delta$ (in our application δ is about $\frac{1}{6}\, l$). Furthermore, the search space may be reduced when there is some intuition of the maximum number of errors expected in a read. If no more than Δ errors are expected in a read, paths of score greater than Δ may be ignored.

In a high-coverage project, a read covering a high multiplicity repeat may contain errors that are not corrected using the spectrum G_l^M. Rather than changing G_l^M for repeat regions, we note that the nucleotide alphabet is small, and the multiplicity of consecutive l-tuples generally does not decrease rapidly. An example is the case of an independent identical distribution of nucleotides immediately after a repeat region. To account for this, we let $\text{score}^u(u, i, a)$ be the minimum edit distance from s to all

T-strings ending in a that contain u and do not contain adjacent l-tuples with a ratio of multiplicities less than ρ . Since this is a rare case, we generally set ρ to 0.01.

A read with minimum score less than Δ is replaced by the corresponding minimum score T -string. If it is not possible to transform a read into a T-string in under Δ changes, it is considered unfixable, and no changes to the read are made. Likewise, it is ambiguous if there are multiple paths with the same cost, and such reads are also considered unfixable. Finally, it is difficult to determine if changes made near the ends of reads are correct, and so reads with erroneous edges are trimmed. After error correction is applied to the set of reads, G_l^M will change. The spectrum G_l^M may be updated and used iteratively to correct reads that were unfixable in a previous iteration. Reads that remain unfixable in the last iteration are discarded.

In order to test our error correction algorithm, simulated reads from sequenced genomes were corrupted with errors and subsequently treated with error correction. Two different error models were used, uniform error distribution, and homopolymeric errors. In the former, errors are distributed uniformly among positions in a read at a specified rate. Errors are distributed randomly between insertions, deletions and mutations, with rates ranging over 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5 and 5%. In Pyrosequencing, there may be difficulty in sequencing homopolymeric regions [118] of > 56 nt, and the latter error model attempts to simulate this. Homopolymeric regions longer than 6 nt were either increased or decreased in length randomly, with a bias toward a decrease.

The results of running error correction on simulated reads at various error rates are presented in Table 4.2. An additional 'fictitious' genome was created to show the operation of error correction on random genomes. The fictitious genome is a 50 kb randomly generated sequence, with 28 exact repeats ranging from 80 to 1000 bases. Exact repeats were used rather than inexact ones, so that the l-tuple multiplicities of repeats were greatly enhanced over random regions. Errors in such regions could often only be detected using the multiplicity ratio threshold. Our error correction method is more effective on the random dataset. Upon inspection of the error-corrected reads for the BACs, many of the errors that were unfixed or corrupted were located in regions of low complexity. In high coverage projects, the correct modifications to make are further obscured in low-complexity regions because many erroneous l-tuples appear with

high multiplicity. Also, many of the corruptions in low-complexity regions were in homopolymeric regions, unfortunately the regions that are most likely to be missequenced in sequencing-by-synthesis. We believe that this problem can be fixed by changing the scoring model to accommodate a bias toward indels within homopolymeric regions. However, such a change will be postponed until a true error distribution of this sequencing technology is known.

While this method works well on short reads, it does not scale well to longer reads. The expected number of errors per read increases with read length, and consequently the size of the search space increases. For high error rates on long reads the size of the search space becomes prohibitive.

Table: 4.2 The result of running error correction on erroneous reads. For each genome the following are reported: *i*, the number of errors in kb initially found prior to error correction in reads retained after error correction, *r*, the number of remaining (unfixed errors), in bases, *c*, the the number of data corruptions (nucleotides that were correct in the original read and incorrect in the fixed read), in bases *%*: the percentage of reads retained (fixable reads) after error correction.

Simulated error rate	Fictitious				85H09				47A01			
	i	r	c	%	i	r	c	%	i	r	c	%
0.5	10.2 K	54	37	99.2	302 K	354	277	99.5	35.3 K	907	717	99.6
1.0	30.7 K	46	17	99.2	603 K	948	754	99.4	70.4 K	1658	1276	99.4
1.5	20.6 K	90	37	98.7	892 K	1364	1080	98.8	104 K	2831	2348	98.9
2.0	40.0 K	123	35	97.5	117 K	2255	1836	97.9	137 K	3735	3089	98.0
2.5	48.7 K	180	77	96.2	141 K	3129	2524	95.8	165 K	5278	4376	96.2
3.0	55.8 K	233	106	92.3	162 K	3884	3154	93.3	190 K	8138	6759	93.8
3.5	60.6 K	256	101	88.8	175 K	4910	3942	88.8	206 K	9593	8009	89.3
4.0	62.0 K	280	132	82.2	180 K	6468	5353	82.5	213 K	10511	8791	83.2
4.5	60.4 K	332	137	73.7	174 K	5797	4775	73.2	209 K	12149	10225	74.8
5.0	57.2 K	353	156	64.7	157 K	5942	4832	61.2	187 K	12474	10512	62.6
random walk	17.3 K	274	223	99.3	55.9 K	5846	4144	99.6	58.9 K	4148	3039	93.9

Table: 4.3 Number of contigs for assembled reads at increasing error rates. Genomes were sampled with 30X coverage with read a read length of 120 ± 40.

Simulated error rate	Fictitious	85H09	47A01
0.5	29	29	47
1.0	27	25	67
1.5	29	39	69
2.0	24	38	84
2.5	32	86	122
3.0	37	74	225
3.5	41	122	72
4.0	56	184	200
4.5	74	183	116
5.0	84	355	460
random walk	28	41	189

After error correction was performed on the test data, the retained (corrected) reads were assembled using a new version of EULER [105]. This version can detect some of the topological features that are added to the de Bruijn graph when there are sequencing inconsistencies (errors or a heterozygous sample) that otherwise fragment the repeat graph. The errors and corruptions after error correction complicate the repeat graph. To obtain 95% coverage for BAC 85H09, 29 contigs (38% of the total contigs) were required at a 3.0% simulated error rate. BAC 47A01 contained more difficult to fix low complexity regions, and required 109 (48% of total) contigs for 95% coverage. With a very low error rate, 0.5%, BAC 85H09 required only 8 contigs for 95% coverage. The results of assembly are shown in Table 4.3.

4.F Conclusions

In this chapter we have examined the feasibility of assembling reads produced by emerging sequencing technologies. While many large contigs may be assembled using even very short reads, substantial (if not prohibitive) finishing efforts are required to entirely resolve all but the simplest of genomes. This is especially true in the assembly of reads that have high error rates that may be a feature of some of these technologies.

Finally, we note that a powerful tool used in current sequencing projects is

the use of mate-pairs for repeat resolution ([147],[104]). The Eulerian approach to fragment assembly is well suited for mate-pairs analysis [104] and it is easy to incorporate mate-pairs in our algorithm for short read assembly. However, since some of the new technologies aim to produce genomes using clone-free data, we did not use mate-pairs in the existing version of the assembly algorithm. The resolution described by repeat graphs in table 4.1 dictate that an extra experimental step is required to finish a sequencing project. It remains a challenge for molecular biologists to determine a high throughput method for producing data similar to mate-pair information, without the tedious task of producing a clone library. We emphasize, however, that due to extreme complexity of the repeat graph for short reads, there may be a need for more accurate insert distance estimates, and it presents yet another challenge for developers of short read sequencing. Alternatively, the repeat structure that is assembled after equivalent transformation can provide information for a high throughput mapping technology. We are eager to see the progress of this field.

4.G Acknowledgements

We would like to thank Drs. Mostafa Ronaghi, Lei Du, and Dirk van den Boom for useful comments regarding new sequencing technologies.

5

Short Read Fragment Assembly of Bacterial Genomes

In the last year, high-throughput sequencing technologies have progressed from proof-of-concept to production quality. While these methods produce high-quality reads, they have yet to produce reads comparable in length to Sanger-based sequencing. Current fragment assembly algorithms have been implemented and optimized for mate-paired Sanger-based reads, and thus do not perform well on short reads produced by short-read technologies. In this chapter a new Eulerian assembler is presented that generates nearly optimal short read assemblies of bacterial genomes and describe an approach to assemble reads in the case of the popular hybrid protocol when short and long Sanger-based reads are combined together.

5.A Introduction

In 2005 the first successful whole-genome shotgun sequencing project performed without using Sanger-based sequencing was completed [79]. The analytical method, pyrosequencing, was proposed in 1998 [119], and has been incorporated into an industrial sequencing platform by 454 Life Sciences (`www.454.com`). While this was one of the original technologies for high-throughput sequencing, many other next-generation competing platforms have recently emerged including the Solexa/Illumina 1G SOLiD Sequencing (`www.appliedbiosystems.com`) that have made their data available to many research

groups. Other recent start-ups, including Helicos (`www.helicosbio.com`) and Complete Genomics (`www.completegenomics.com`) plan to release their sequencing data soon (see [83] for an overview of emerging short read technologies). Although each technology is able to produce vast quantities of sequence information, in every case the underlying chemistry limits reads to very short lengths in comparison to Sanger-based reads: from over hundred bases (454 Life Sciences) down to as little as 24 (Solexa/Illumina and Applied Biosystems). These technologies usher a new era of High Throughput Short Read (HTSR) sequencing.

The applications of HTSR sequencing have been largely limited, albeit with great success, to resequencing [2], gene expression [70], and genomic profiling [7], which all require a reference genome. Although many model organisms have been sequenced, the success of recent comparative genomics studies [30] will increase the demand for *de novo* assembly, which will be assisted by the assembly of HTSR sequences. Furthermore, the discovery of rampant structural polymorphisms in the human genome [41] indicates that it is necessary to perform *de novo* assembly of even the same species in order to capture all variations.

Even before the release of the first draft bacterial genome assembly by 454 Life Sciences, we published a paper examining *de novo* assembly of short reads [23] (see also Whiteford et al., 2005 [148]). In the absence of available data we simulated reads from bacterial genomes with lengths and error-rates of what the contemporary proof-of-concept short-assembly papers had cited. We came to the conclusions that while the high-coverage facilitate the error correction in reads prior to assembly, it will be difficult to assemble very large contigs without mate-pair information. Recently, 454 Life Sciences developed Newbler assembler that is now a part of the software package distributed with 454 sequencing machines. Newbler is an excellent assembler that takes into account the specifics of pyrosequencing errors to generate accurate contigs.

The publication [79] has shown that short read assembly is indeed difficult; the assembly of *Streptococcus pneumoniae* (*S. pneumoniae*) had 255 long contigs (i.e., contigs longer than 500 bp), of which 11 were misassembled. Later, the proprietary 454 Newbler assembler was improved and it now produces 253 long contigs (of total length 2000 Kb) and no missassembled contigs. The *S. pneumoniae* genome contains

167 exact (unresolvable) repeats at least 120 bases long[1] that fragment the assembly into 504 contigs, of which 136 are longer than 500 bp. The difference between 253 and 136 illustrates the gap between the performance of the existing short read assemblers and the lower bound on the number of contigs an *optimal* assembler would produce. Our new assembler, EULER-SR, produces only 127 long contigs of total length 2001 Kb thus illustrating that it comes close to the "optimal" assembler (136 long contigs of total length 2091 Kb).

The assembly of short-reads is delayed by the inability to adapt older assembly techniques from previous sequencing projects. Many whole genome assemblers are optimized for assembling the mate-paired long reads that until recently have been the sole type of data produced by large-scale sequencing centers. Programs such as Phrap (`www.phrap.org`) and PCAP [57] simply do not run on next-generation data, and Arachne [61] produces an over-fragmented output in the absence of mate-pairs. Similarly, EULER+ assembler [105] does not scale well for high-coverage short read sequencing projects.

The Eulerian approach for assembly was proposed by Pevzner, 1989 [100] in the context of Sequencing by Hybridization (SBH), which may be viewed as the very first short read sequencing technology.[2] In a pioneering paper, Idury and Waterman, 1995 [59] extended the Eulerian approach to Sanger reads. Briefly, in Eulerian assembly, a *de Bruijn* graph is created with a vertex v for every l-tuple present in a set of reads, and an edge $(v,\ v')$ is created if an $(l+1)$-tuple exists in the reads that contains v as an l-prefix and v' as an l-suffix. An assembly corresponds to an Eulerian path [31] through this graph. However, the practical applications of both [103] and [59] were hindered by high error rates of SBH and Sanger sequencing (circa 1995) that made the direct applications of the Eulerian approach difficult. In the Eulerian assembly described in [106], sequencing errors create extra vertices and edges in this graph that require *error correction* procedures to improve the assembly quality.

[1] Most reads in this study are shorter than 120 bases.

[2] The paper [100] illustrates some potential advantages of the Eulerian path approach over the "overlap-layout-consensus"' approach for fragment assembly. For example, while [103] described a simple algorithm for constructing the SBH repeat graph, it is not immediately clear how to generalize the approaches in [57, 61] for efficient construction of the repeat graph even in the simple case of SBH "reads"'.

More recently, Pevzner et al., 2004 [105] proposed a new Eulerian assembler based on the notion of *A-Bruijn graphs*. The EULER+ assembler [105] deals with errors in reads by inducing vertices with ungapped alignments that allow mismatches, rather than the exact l-tuples in de Bruijn assembly, and by defining a set of graph-simplification techniques that remove erroneous edges. Large benchmarking experiments in [105] demonstrated that the A-Bruijn approach compares well with Phrap and Arachne in assembling BACs and bacterial genomes. A similar approach was recently described by Myers, 2005 [91] and Medvedev et al., 2007 [82]. The A-Bruijn graph was recently used in various applications including multiple alignment [114], finding composite repeats [153], shotgun protein sequencing [6], and dissecting the history of segmental duplications in human genome [64]. However, the application of the *A*-Bruijn based assembly for short read sequencing (with much higher coverage than in typical long-read sequencing projects) is difficult since the graph structure that is used scales with coverage and thus demands a large amount of memory. Therefore, the approach in [23] has to be modified for high-coverage (and thus memory-intensive) short read assembly.

In this chapter we describe how to modify the A-Bruijn technique from [105] for short read assembly and present a memory-efficient de Bruijn-based approach that supports A-Bruijn-like graph-correction operations. Our EULER-SR assembler substitutes the maximum Spanning Tree optimization of the A-Bruijn graph by the Maximum Branching optimization on de Bruijn graphs. We benchmark our new EULER-SR assembler on data generated by two 454 bacterial sequencing projects (with an average read length of $\approx$ 100 bp). Our assembly of 454 reads compares well to the proprietary 454 assembly tool Newbler, and is the only assembler tool currently capable of assembling shorter reads such as those produced by the Solexa (we were able to produce nearly optimal BAC assembly with Solexa reads). In a difference from the Newbler assembler that outputs the set of unrelated contigs, EULER-SR outputs both the contigs and the *repeat graph* [106] of the assembled genome thus connecting the contigs by repeats and directing finishing effort. In addition, this facilitates the use of mate-pairs when they become available as recently announced by 454 Life Sciences and Illumina.

Currently, most sequencing centers utilize 454 reads by combining them with low-coverage Sanger-based reads for bacterial assembly and finishing [47]. They first

assemble only 454 reads using Newbler and later transform long Newbler contigs (longer than 500 bp) into overlapping pseudo-reads modeling typical Sanger reads. These pseudo-reads are further combined with real Sanger reads and subjected to a conventional assembly with programs like Arachne. This computational protocol is clearly not optimal: for example, even if Newbler was substituted by an "ideal" assembler, the valuable information about short contigs (shorter than 500 bp) would be lost. In particular, all valuable information about repeated regions shorter than 500 bp (covered by 454 reads) is not utilized in this hybrid approach. However, since Newbler is not designed to handle mate-pairs and since Arachne is not designed to handle short reads, no better solution is currently available. Below we show that EULER-SR is well suited for hybrid 454+Sanger approach to assembly and finishing.

5.B Results

We analyzed 454 reads from *E. coli* and *S. pneumoniae* (454 Life Sciences), and Solexa reads from a human BAC[3] (Solexa). Many HTSR sequencers produce raw data that is different from the traditional read traces found in Sanger sequencing, such as the *flowgram* reads produced by 454 or *color-space* reads produced by Applied Biosystems. It is possible to take advantage of the additional information in the raw read information, as done by Newbler assembler (454 Life Sciences), however EULER-SR solely uses reads and quality files (if available) so that it is applicable to any sequencing technology. Below we demonstrate that EULER-SR compares well with 454 Newbler assembler and generates nearly optimal assembly of Solexa reads despite the fact it does not take advantage of the peculiarities of the platform-dependent error models.

5.B.1 Error correction

Our approach begins by preprocessing reads to remove errors. When quality values are available, we trim or discard low quality reads, however typically our error correction routine will detect low quality reads without quality values. Our method for error correction, as described in Methods, introduces the fewest changes possible in a

[3] *E.coli* is 4,639,675 bp long, *S. pneumoniae* is 2,160,837 bp long and BAC is 173,427 bp long (chromosome 6, bases 30537344-30710771).

Table: 5.1 The results of error correction on 454 GS20 reads. Error correction using Spectral Alignment was applied to each set of reads after trimming ends of reads with Phred scores averaging less than 20 over 5 bases. The number of reads sequenced, number of quality-trimmed reads, and the number of corrected reads after error correction, along with the number of reads that have an exact match in the genome is given. Reads were mapped to the original genome by querying a compressed suffix array as described in [76].

	Reads, in thousands		
Genome	Total (error-free)	Quality trimmed (error-free)	Error corrected (error-free)
E. coli	1113 (757)	1113 (765)	1012 (1008)
S. pneumoniae	1063 (605)	1063 (652)	963 (954)

read such that every l-tuple in the read belongs to a set of l-tuples called a *spectrum*, $\mathcal{S}$, or considers the read unfixable if too many changes are required. We iteratively apply our error correction as described in Methods to a set of reads, first setting $\mathcal{S}$ so that reads in regions of high coverage are corrected, and gradually increase the set of reads in which we fix errors by expanding the set S.

We tuned the parameters for iteratively fixing errors on 454 *E. coli* reads. This data set has approximately 1.1 M reads for 27-fold coverage of the genome. The reads are relatively high quality on average, and only 0.5% of bases were trimmed using quality scores. We measure the rate of error correction by measuring the number of error-free reads, i.e, reads that appear exactly in the genome (68% of the reads error-free). Our error correction routine divides the reads into two sets: a large set of *corrected* reads and a small set of *non-fixable* reads. The non-fixable set has $\approx 0.1M$ reads (9% of all reads), and of the remaining corrected set, 99.6% reads are error-free (Table 5.1). Applying our error correction to the *S. pneumoniae* (without changing parameters) resulted in slightly lower (99.1%) but still high rate of error correction.

The bulk of EULER-SR time is taken by error correction; it takes ≈ 12 hours to perform error correction on a single, 1.8 GHz workstation and only $\approx 1/2$ hour to assemble the genome. Fortunately error correction represents an embarrassingly parallel application, and only takes a matter of minutes on a modest cluster. EULER-SR provides support for parallelizing error correction on the SGE cluster platform.

5.B.2 Fragment assembly of short reads

After correcting errors we ran EULER-SR on the set of corrected reads. Our assembly procedure is performed in a set of steps that first builds the de Bruijn graph, corrects errors in the graph, and transforms paths in the graph into assembled sequences. When constructing the de Bruijn graph the l-tuple size should be chosen to be large enough to avoid excessive tangling of the graph that leads to shorter contigs, yet small enough to not fragment the genome since every $(l+1)$-tuple must be covered by a read in the de Bruijn graph. We found that $l = 35$ removes most tangles at a cost of slightly fragmenting the assembly. To resolve the fragmentation, we 'artificially' connect source and sink edges that share a smaller unique overlap at the last stage of the algorithm. We combined the error-corrected reads with the non-fixable reads that were discarded during error correction in order to assemble low-coverage regions.

The parameters for graph correction control the maximum girth (for bulge and whirl removal), maximum source or sink edge length to delete during erosion, and minimum edge multiplicity to detect chimeric edges. We tuned our parameters for assembling *E. coli*, and used this to assemble *S. pneumoniae*. After error correction, all computations were performed on a desktop computer. We compared our assembly with ones produced by the 454 Newbler assembler (Table 5.2). For each assembly we report the number of long contigs (i.e., contigs longer than 500 bp), the total length of long contigs, the N50 contig size (the size of the contig such that 50% of the assembly is contained in contigs of size N50 or greater), and the coverage using long contigs. Each of these statistics in isolation may be misleading, for example, an assembler that misjoins contigs may have the best parameters. However, neither the improved version of Newbler, nor EULER-SR produced misjoined contigs making these parameters comparable for different assemblers[4].

The theoretical limits of short read assembly may be evaluated by constructing the repeat graph of a genome with a minimum repeat length equal to the typical length of a read. The repeat graph corresponds to the condensed de Bruijn graph if an error-free read is generated at each position in the genome. Therefore, the repeat

[4]As described above EULER-SR does not try to estimate the multiplicities of tandem repeats and misses three copies of tandem repeats in *E. coli* (approximately 1600, 1000, and 300 nucleotides) and one copy of a tandem repeat in *S. pneumoniae* (approximately 500 nucleotides).

graph models an "optimal" assembler that can be benchmarked against the existing assemblers (Table 5.2). The optimal assembly of *E. coli* on reads of length 120 has 800 contigs with 94 long (i.e., longer than 500 bases) contigs covering 4560 Kb. The optimal assembly of *S. pneumoniae* on reads of length 120 has 504 contigs with 136 long contigs covering 2091 Kb. For *S. pneumoniae*, EULER-SR comes close to optimal assembler with 127 long contigs covering 2001 Kb. Newbler generated twice the number of larger contigs (253) with similar total size (2000 Kb). We emphasize that Newbler uses the platform-dependent error model of 454 technology and has access to *flowgrams* reflecting the specifics of pyrosequencing errors. This may explain the deterioration in EULER-SR performance in the case of *E. coli* (199 long contigs covering 4277 Kb) as compared to Newbler (141 long contigs covering 4531 Kb). When Newbler is run on a FASTA data that is the equivalent of only nucleotide sequences (an abuse of the intended usage), its performance degrades to 311 long contigs covering 4,523Kb. This experiment demonstrates that Newbler is an excellent tool for assembling 454 reads (as it uses flow values to mitigate sequencing errors) that may not be easy to modify for other sequencing technologies. As 454 read quality is constantly improving, the additional data provided by 454 flowgrams is becoming less crucial. The difference in the errors rates between different sequencing projects undertaken at different times/labs may explain a somewhat better EULER-SR performance on *S. pneumoniae* as compared to *E.coli*.

We have also assembled a human BAC using $\approx$ 1 million 27 nucleotide long Solexa reads generated at the Joe Ecker lab at the Salk Institute (150X coverage). The Solexa technology is currently being optimized and the error model of the Solexa platform remains poorly understood. In particular, the error rates of Solexa reads are position-dependent and highly variable across different machines and even across different runs of the same machine (Alla Lapudus, personal communication). As a result, while assembly of Solexa-sized simulated reads was analyzed in [146], the assembly of real Solexa reads proved to be more challenging. Since Solexa reads have very few indels, we have chosen to use the heuristic error correction similar to [106] instead of Spectral Alignment. This heuristic iteratively finds the mutation in a read that makes the most l-tuples *solid* (see Methods) until all tuples in the read are solid, or no mutation is found to make an l-tuple solid (we use $l = 15$). The error correction partitions the reads into 592K of

Table: 5.2 Summary of bacterial assemblies using 454 reads. We report assemblies using EULER-SR and Newbler, and the repeat graph as a reference for the optimal assembly. #Long contigsContigs: the number of contigs greater than 500 bases. Total length of long contigs: the sum of the length of all long contigs. N50: the size of the smallest contig such that 50% of the length of the genome is contained in contigs of size N50 or greater.

Genome	Assembler	# Long contigs	Total length of long contigs (in Kb)	Total coverage	N50
S. pneumoniae	EULER-SR	127	2001	32619	
	Newbler	253	2000	11905	
	Repeat graph	136	2091	36004	
E. coli	EULER-SR	199	4277	46887	
	Newbler	141	4531	60757	
	Repeat graph	94	4560	125693	

error-corrected reads and 402K of non-fixable reads. The significantly higher proportion of non-fixable reads (as compared to 454 platform) reflects the higher error rates of Solexa reads and some surprising run-dependent biases in the distribution of errors along the reads. 580K out of 598K error-corrected reads are error free, 97% of all error-corrected reads (as compared to over 99% in case of 454 reads). The assembly results in 82 long contigs (larger than 500 bp) of total length 100,360. The optimal assembly (as revealed by the repeat graph on 24-mers) results in 71 long contigs of total length 124,660.

5.B.3 Combining short read assembly with mate-pair data

One of the goals of HTSR sequencing is to sequence reads without the expensive and slow step of producing a clone library. While it may be difficult to produce paired reads in the traditional sense of sequencing opposite directions of an amplified clone, it is possible to amplify and sequence short paired-end tags in parallel. To construct the paired tags, short genomic sequences are circularized using a known linker sequence, and are cleaved outside the ligation site using Type IIs restriction enzymes leaving 25-31 base tags of genomic sequence available for sequencing [132]. While other technologies are also capable of producing longer paired ends (Michael Egholm, personal communication) there are currently no publicly available mate-paired short read data sets for these genomes. We therefore decided to simulate the mate-pair information to investigate how

it improves the assembly.

We modified the mate-pair assembler, Euler-DB [104] to run on the volume of data generated in HSTR sequencing and tested it on simulated paired-end data of short reads. The reads for *E. coli* and *S. pneumoniae* were mapped to the reference genomes, and tag libraries of sizes 2.5Kb, and 10Kb were generated by selecting 50 bp tags from the reads separated by 2,500±250 and 10,000±1000 bases, for each library. The number of tags generated (300,000) is less than the total sequencing capacity of a typical HSTR sequencing run. The results of our assembly using mate-pairs are listed in Table 5.3[5].

The current practice in most sequencing centers is to use low coverage Sanger sequencing to complement short reads in hierarchical assembly [47]. Because EULER-SR may be used with any length read, we simulated paired-end Sanger sequences for *S. pneumoniae*, and performed assembly by combined these reads with *S. pneumoniae* 454 reads. The results are shown in Figure 5.1. While adding long reads results in substantial improvement in the assembly quality, we found that remarkably small number of long reads contribute to this improvement. For example, in the case of 5X read coverage by long reads, $\approx 96\%$ of all mate-pairs maped to a single contig composed of 454 reads and thus do not provide complementary information for 454-based data except for marginal potential improvement in the consensus quality. Moreover, most mate-pairs mapped to multiple contigs span only two edges in the de Bruijn graph and thus also do not improve the assembly of 454 reads. As a result, only 238 out of ≈ 6800 long mate-paired reads (only $\approx 3\%$) map to more than two edges in the de Bruijn graph[6] and thus meaningfully contribute to improving the assembly.

Our analysis of the hybrid protocol revealed that the overwhelming majority of long reads do not improve upon the assembly already generated from short reads. It does not necessarily mean that the coverage by long reads in hybrid approaches may be reduced without affecting the assembly quality: for example, Figure 5.1 does not reflect

[5]The surprising deterioration of N50 statistics for 10 Kb spacing (as comparing to 2.5 Kb spacing) reflects ambiguities in mapping longer paths between mate-pairs in highly tangled de Bruijn graphs.

[6]123, 72, 31, 9, and 3 mate-pairs map to 3, 4, 5, 6, and more than 6 edges correspondingly.

Table: 5.3 Summary of the assemblies using real 454 reads combined with simulated mate-pairs. Each end of a clone is 50 bases.

Genome	Tag library	# Long contigs	Total length of long contigs (in Kb)	N50
S. pneumoniae	2.5Kb	156	2111	43418
	10Kb	139	2224	36253
E. coli	2.5 Kb	163	5016	91252
	10 Kb	126	4897	101647

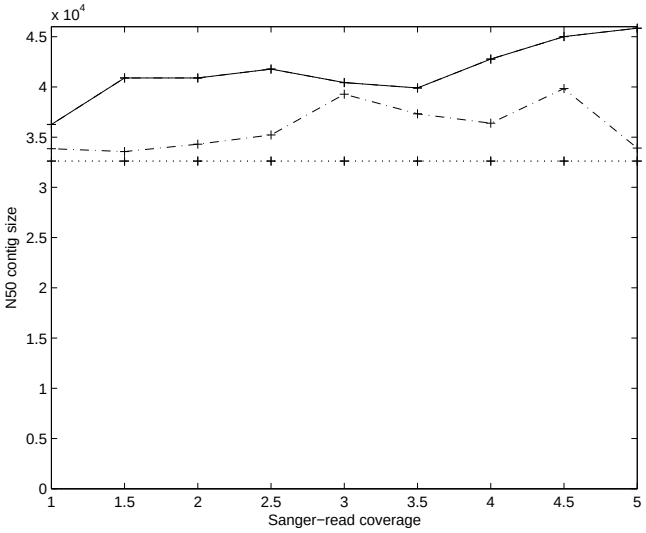


Figure 5.1 Assembly of *S. pneumoniae* using 454 reads complemented with simulated Sanger reads (3000 $\pm$ 300 bp spacing). Uniformly distributed 800 base reads were generated for coverage varying from 1X to 5X, and then assembled using EULER-SR. We show the assembly using mate-pairs (solid), assembly using unpaired Sanger reads (dashed), and the base assembly quality of 454 reads (dotted).

the contribution of long reads to improving scaffolds. However, Figure 5.1 suggests that a critical review of the benefits of the hybrid approaches may be necessary to compare the relative benefits and costs of generating shotgun long reads as compared to other finishing approaches.

5.C Discussion

We presented the EULER-SR short read assembler that significantly reduces the memory requirements of previously described Eulerian assemblers and has a potential to produce nearly optimal assemblies of bacterial genomes using modest computational resources. Our assembly software is available at `http://euler-assembler.ucsd.edu`.

We have used assemblies produced by the proprietary Newbler assembler for benchmarks, as there are currently no other methods available for assembling short reads. Newbler is an excellent tool for assembling 454 reads as it uses raw flow values to mitigate sequencing errors. Our assembler compares well with Newbler yet does not rely on proprietary information, allowing it to assemble reads from multiple sequencing platforms.

Sundquist et al., 2007 [138] recently proposed a new SHRAP experimental protocol that may enable the assembly of mammalian genomes with short reads. SHRAP uses the older version of EULER [105] as an engine for assembly of BAC-sized genome segments, which are then joined. One benefit of EULER-SR assembly is that assemblies of large genomes may be performed because de Bruijn graph construction is intrinsically parallelizable. The de Bruijn graph of the union of two sets of reads is simply the union of the de Bruijn graph for each independent set of reads. Therefore it may be possible to generate short-read assemblies on large genomes by simply taking the union of small sets of reads succinctly represented as condensed de Bruijn graphs.

5.D Methods

5.D.1 Error correction in reads

Our assembly proceeds in several steps: error correction, graph construction, graph correction, and assembly by transforming paths in the corrected graph into contigs. Base miscalls and indels are inevitable in sequencing projects and it is necessary to detect errors in reads in order to accurately determine the finished sequence. In the past this was done after assembly by mapping reads to the consensus sequence. Pevzner et al., 2001 [106] introduced error correction before the assembly and demonstrated that it greatly simplifies the assembly. Now this is done as a standard preprocessing step before assembly using a multiple alignment of reads [140], or by clustering and aligning reads prior to assembly [11]. In HTSR sequencing, constructing multiple alignments of short reads is very time consuming, and so we correct errors in reads prior to assembly using a method called *Spectral Alignment* (SA) that does not use a multiple sequence alignment. This method takes a read r and a set of l-tuples $\mathcal{S}$, called a *spectrum*, and finds the minimum number of substitutions, insertions, and deletions in r required to make every l-tuple in r belong to $\mathcal{S}$. The set $\mathcal{S}$ is chosen by counting the frequency of all l-tuples present in all reads in a sequencing project, and selecting tuples that occur with multiplicity above some threshold m (called *solid* l-tuples). An iterative solution to SA was described in [106], followed by a dynamic programming solution in [23]. The dynamic programming solution to SA allows for efficient searching through insertions and deletions, and so it is particularly well-suited for fixing errors in reads generated by pyrosequencing, in which the errors are biased towards indels. The Spectral Alignment approach implemented in EULER-SR is faster and has a higher rate of error correction than the approach described in [23].

The choice of appropriate parameters for fixing errors using *Spectral Alignment* should minimize the number of erroneous l-tuples remaining in reads while not affecting correct but low coverage sequences. Consider a sequencing project that produces N reads of average length L over a genome of length G. The average number of reads covering an l-tuple is $a = N \times (L - l)/G$ and is approximately distributed by a Poisson with parameter a. To minimize the number of correct sequences considered to be erroneous

we pick a multiplicity m for which there are few expected l-tuples covered by less than m reads. The probability an l-tuple in a genome is covered m times is $\sum_{y=0}^{m} P_Y(y)$, where $Y \sim$ Poisson with parameter a. For a typical high-throughput sequencing project of a bacterial genome of 4 million bases with 1 million 100 bp long reads, there are under 100 20-mers expected to be covered less than 5 times. Furthermore, under the gross simplification that the sequences are random, and there is a 1% random error rate, no erroneous l-tuples are expected to be made solid by coincident errors.

5.D.2 de Bruijn graph construction

Due to the massive scale of the data in short read assembly, it is necessary to perform assembly in linear or close to linear time. Although it is possible to construct a de Bruijn graph in linear time, memory efficiency is more of a premium for short reads assembly than run time. Our de Bruijn graph construction is implemented in several stages, in some of which we trade linear time for methods that run on sorted lists (thus $O(n \cdot logn)$ time) but reduce memory requirements.

For a genome of length L, the de Bruijn graph has $O(L)$ vertices and $O(L)$ edges, regardless of the number of reads in the data set. We find the set of vertices in the de Bruijn graph using an efficient hashing structure in linear time proportional to the number of reads R in the data set. We then represent the vertices as a sorted list V of tuples allowing us to discard the memory overhead of the hashing structure required for O(1) time access. We next generate adjacencies by querying the vertex list V with a binary search for every adjacent pair of l-tuples in the set of reads R. By representing the graph as an adjacency list, our de Bruijn graph uses a maximum of $O(L)$ * $(k+l)$ bytes, where k is the memory allocated for each vertex (40 bytes in the current implementation). For subsequent phases we form a *condensed de Bruijn graph*, where every simple path consisting of vertices of in-degree and out-degree 1 is substituted by a single (labeled) edge. Because the reads were used to generate the de Bruijn (and thus the condensed) graph, every $(l+1)$-tuple in a read maps to a unique edge and position in the condensed de Bruijn graph. We use this to define a path of edges for each read, and to assign a weight to every edge equal to the number of reads mapped to the edge. These paths are used after graph correction to find the Eulerian path through the de

Bruijn graph that corresponds to the assembled genome.

5.D.3 de Bruijn graph correction

If reads covered every $(l+1)$-tuple in the genome and were error-free, the generated de Bruijn graph G^* would represent the *repeat graph* of the genome with minimal repeat length $l+1$. However after error correction, a small number of errors remain resulting in some added and removed edges as compared to the repeat graph. For example, a single mutation in a read will create $(l+1)$ extra edges in the de Bruijn graph. Our goal is to construct the graph G^* given G. If we assume every $(l+1)$-tuple in the original genome is sequenced correctly by at least one read, then G^* is a subset of the vertices and edges of the graph on real reads G. Thus we may perform additional error correction by detecting and removing erroneous edges on the graph G.

We adapt the methods for *graph simplification* on A-Bruijn graphs from [105] to perform *graph correction* on de Bruijn graphs. We distinguish the two methods because graph simplification aims to capture the repeat-consensus and mosaic structure of a genome, while the goal of graph correction is to simplify the de Bruijn graph until all erroneous edges are removed, and no further. For example, in graph simplification, repeats with high similarity are typically merged into a single edge corresponding to the the repeat consensus sequence. A corrected de Bruijn graph contains a separate path for each distinct repeat sequence.

When an A-Bruijn graph is constructed from a set of pairwise alignments, gaps and substitutions in an alignment create undirected cycles called *bulges*, and inconsistencies in an alignment create directed cycles called *whirls* [105]. A cycle is called *short* if its girth is less than a fixed parameter g. The algorithm to generate an A-Bruijn graph employs a heuristic to find a graph that represents the largest possible set of consistent alignments through solving the Maximum Subgraph with Large Girth (MSLG) problem [105]. Because this problem is hard for arbitrary girth values, a heuristic is used that replaces the A-Bruijn graph by its Maximum Spanning Tree, and successively adds edges from the original graph to the spanning tree if they do not create a cycle of girth less than g.

A similar approach for removing short cycles is used for fragment assembly on

the de Bruijn graphs. One property of a de Bruijn graph is that every vertex should be reachable by a directed path from the source vertex, if only one source vertex exists. A shortfall of the bulge removal method in A-Bruijn graphs is that when a de Bruijn graph is replaced by its undirected maximum spanning tree, certain vertices may not be reachable from the source by directed paths. Although a *path straightening* heuristic [105] is used in the A-Bruijn graph construction to repair edges to unreachable vertices, we circumvent this problem entirely by replacing the de Bruijn graph with its Maximum Branching [37, 26], using the method described in [46] (the branchings have directed paths from the source vertex to all vertices). Because "alignments" are found at each vertex by perfectly matching sequences of length l, directed cycles are rarely erroneous other than those covering long low complexity regions, e.g. homopolymeric and dinucleotides. When checking to see if adding an edge to the maximum branching forms a cycle, we distinguish between directed and undirected cycles, and add an edge to the branching if it does not create an undirected cycle of girth u, nor a directed cycle of length d.

While tandem repeats can be detected as whirls in the de Bruijn graph (typically directed cycles with two edges as shown in Figure 5.2, *top*), finding the number of copies (multiplicity) in a tandem repeat is a difficult problem for any assembler. For example, the multiplicity of perfect tandem repeats of length longer than half of the read length cannot be inferred from de Bruijn graphs. As a result, inferring the multiplicities of tandem repeats usually amounts to error-prone coverage analysis (e.g., a tandem repeat with multiplicity 3 is expected to have 50% more coverage than a repeat with multiplicity two). To avoid potential errors caused by the coverage analysis (short read technologies often produce uneven coverage) and to reduce the fragmentation due to tandem repeats, we assume there are only two copies of the perfect tandem repeat, and construct the assembly as a path that traverses the repeat twice, as shown in Figure 5.2, *bottom*. While this procedure may underestimate the copy number of tandem repeats we found that it leads to very few errors[7]. We therefore feel that it is a practical approach, at least until more information about coverage and specifics of errors of short read technologies becomes available.

[7]Also in cases where coverage across the tandem repeat is low, tandem repeats may be collapsed into a single copy.

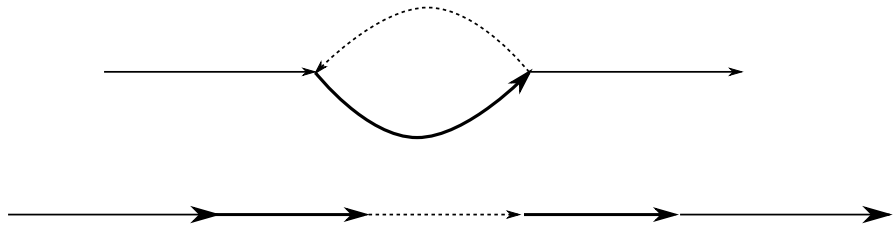


Figure 5.2 A tandem repeat as a whirl consisting of two edges in the condensed de Bruijn graph, and its transformation into a tandem duplication with multiplicity 2. The bold edge is the repeat, and the dashed edge represents the sequence between the consecutive copies of the repeat.

Reads that are mapped to edges that have been deleted during graph correction are threaded through the remaining edges. Let the path of such a read be $(e_0, \ldots, e_i, \ldots, e_m)$, and e_i be the removed edge. We search for an alternative path $P = (a_0 \ldots a_n)$ from the start to the end of edge e_i such that the sequence corresponding to e_i and P are sufficiently similar, and replace e_i with P if such a path is found.

In addition to bulge and whirl removal we also apply two additional techniques for removing erroneous edges: erosion as described in [105], and low weight edge removal. Erosion removes short sources and sinks from the graph. Low-weight edge removal is typically a way to detect low-quality chimeric reads. Although most HSTR sequencers do not use clone libraries, sequencing errors at the ends of reads create erroneous edges similar to those created by chimeric reads.

After the de Bruijn graph has been corrected we apply the equivalent transformation described in [106] to resolve repeats that are shorter than the length of a read. If no mate-pair information exists, we cut paths that do not cover repeats with an operation similar to the x-cut [106]. When mate-pair information is available, we postpone the cutting operation, and apply equivalent transformation with mate-pairs [104].

5.E Acknowledgments

We are indebted to Haixu Tang for much advice that was crucial for successful transformation of EULER+ into EULER-SR. We thank Xiaohui Huang and Eric Roller for many helpful discussions on emerging short read technologies. We thank Michael Egholm and James Knight for providing us with 454 reads and assemblies and helpful

discussions about pyrosequencing reads. We thank Joe Ecker and Ronan O'Malley for providing us with Solexa reads and helpful discussions on BAC resequencing using Solexa platform. Finally, we are grateful to Alla Lapidus and Jeong-Hyeon Choi for advice on fragment assembly. This research is supported by NIH grant 1R21HG004130-01 grant and by NSF Infrastructure grant EIA-0303622.

6

de novo Fragment Assembly with Short Mate-Paired and Error-Prone Reads

Increasing read length is currently viewed as the crucial condition for de novo fragment assembly with next-generation sequencing technologies. However, introducing mate-paired reads (separated by a gap of length *GapLength*) opens a possibility to transform short mate-pairs into long *mate-reads* of length $\approx$ *GapLength* and thus raises the question whether the read length (as opposed to *GapLength*) even matters. A new tool is described, EULER-USR, for assembling mate-paired short reads and use it to answer the question whether the read length matters. We further complement the ongoing *experimental* efforts to maximize read length by a new *computational* approach for increasing the effective read length. While the common practice is to trim the error-prone tails of the reads, we present an approach that substitutes trimming with error correction using repeat graphs. An important and counterintuitive implication of this result is that one may extend sequencing reactions that degrade with length "past their prime" to where the error rate grows above what is normally acceptable for fragment assembly. We demonstrate that this approach allows one to increase the effective length of Illumina reads while making them nearly error-free.

6.A Introduction

The field of high-throughput sequencing has grown recently in both applications and computational support. This is enabled by the many platforms that exist for high-throughput sequencing, including those produced by 454 Life Sciences [80], Illumina 1G Genome Analysis System (`www.illumina.com`), Applied Biosystems SOLiD Sequencing (`www.appliedbiosystems.com`), and Helicos GSS Sequencing (`www.helicosbio.com`) . Although the 454 sequencing platform is now producing reads that are of similar length to Sanger reads, the underlying paradigm is that a higher throughput may be achieved at the sacrifice of read length. The technologies with the highest throughput currently available produce short, 20-40 base reads, distinguished as *ultrashort* reads.

Many recent successful applications of ultrashort reads have used the reference genomes for whole genome re-sequencing [54], chromatin remodeling mapping [127] and whole-genome methylation [7] profiling. An analysis in [148] showed that the number of reads uniquely mapped to the human genome grows with the increase in reads length but reaches a plateau after the first ≈ 40 nucleotides. This implies that there is little incentive to increase the read length in re-sequencing applications since it provides little return on investment. While generating 40 nt reads is usually sufficient for resequencing, de novo fragment assembly may require longer reads. The Illumina platform can easily generate longer reads, but the error rates deteriorate after the first 35 nt making the ends of reads not suitable for fragment assembly. This again provides little incentive to generate longer reads. By extending the effective length of reads, our new EULER-USR assembler generates more reads that span the repeats and thus improves the assemblies.

Most *de novo* assemblers for Sanger reads follow the "overlap-layout-consensus" paradigm that is optimized for such reads [57, 61] and does not scale well for short read assembly. In contrast, most approaches to short read assembly including EULER-SR [24], Velvet [151], and ALLPATHS [20] use an alternative Eulerian approach that model the assembly problem as a search for an Eulerian path in a de Bruijn graph [106].

The advantage of the Eulerian approach is that it generates a theoretically optimal fragment assembly of reads of length k (in high-coverage projects) by essentially mimicking fragment assembly as Sequencing by Hybridization (SBH) problem on a vir-

tual DNA array with all k-mers [103]. Idury and Waterman, 1995 [59] demonstrated how the Eulerian approach for SBH can be applied to fragment assembly of Sanger reads. However, the Eulerian approach works best for error-free reads and quickly deteriorates as soon as the reads have even small number of basecalling errors. To alleviate this limitation, two different *error-correction* approaches are used: error correction in reads prior to assembly [106, 20], and *post-hoc* graph corrections that remove spurious edges from the assembled de Bruijn or A-Bruijn graphs [107, 151].

Correcting errors in reads prior to assembly was shown to be useful for both Sanger and 454 Life Sciences reads [106, 24] (error rates reduced forty-fold from 1.2% to 0.03% for Sanger reads). However, this approach essentially transforms already rather accurate reads ($\approx$ 1% error rate) into nearly error-free reads. The efficiency of the existing error correction approaches quickly deteriorates with even a small increase in the error rates of original reads (e.g., from 1% to 3%). For example, an optimized error correction tool from Tammi et al., 2003 [141] was able to reduce the error rates from 3.4% to 0.3% on simulated data, only a tenfold decrease. While it looks like a reasonably low error rate, it makes it nearly impossible to apply the Eulerian approach that does not tolerate even such seemingly small error rates. This makes the full-length Illumina reads "ineligible" for Eulerian assembly (Figure 6.1 illustrates that the error rate is $\approx$ 20% at the ends of reads) and sets an accuracy bottleneck for developing new sequencing technologies.

In this chapter we focus on the Illumina technology and describe how to increase the "usable" length of error-prone Illumina reads while keeping them nearly error free. Although this chapter limits the benchmarking of EULER-USR to the Illumina technology, our algorithm is applicable to any technology characterized by high error rates, such as the Helicos platform. While the average error rate in Illumina reads is under 2% for the first $\approx$ 30 nucleotides (Figure 6.1), it quickly increases in the tails of the reads reaching $\approx$ 20% at position 50. No existing short read assembly tool can efficiently deal with such high error rates and the conventional wisdom is that the read tails become unusable when the error rate exceeds 3%. Below we introduce an alternative error correction approach that uses the de Bruijn graph constructed on the accurate read prefixes in order to correct the error-prone read suffixes by fitting them into the de Bruijn graph.

We demonstrate that our method improves the quality of assembly without changing the experimental protocol. Since EULER-USR can assemble error-prone reads, we hope that it can catalyze developments of sequencing platforms aimed at generating longer but less accurate sequencing reads.

Reads may be combined with mate-paired information to further improve the quality of assemblies and the next generation sequencing companies are actively pursuing both increasing the read length and effectively generating mate-pairs. For example, Illumina recently increased the effective read length from 35 to 50 nucleotides and announced the plans to provide capabilities for 75-100 nt reads in 2009. On the other hand, Illumina and various sequencing centers are exploring applications of jump and PET libraries [93] for generating mate-pairs in the context of short read technologies. While increasing the read length is a high priority for most next-generation sequencing companies, there exists an opinion that the read length almost does not matter if one uses mate-paired reads. Indeed, Pevzner and Tang, 2001 [104] demonstrated that most *mate-pairs*: "read$_{start}$ - GAP of length d - read$_{end}$" can be transformed into *mate-reads* "read$_{start}$ - SEQUENCE of length d - read$_{end}$" by filling in the gap of length d with the nucleotide sequence representing an appropriate path in the de Bruijn graph. As a result, one can generate contiguous long reads of length $2 \cdot l + d$ (*span* of mate-pairs) from short mate-paired reads of length l making the read length almost irrelevant (typically, $d \gg l$).

We show how EULER-USR utilizes mate-pairs to significantly improve assembly, and further use it to answer the question whether read length matters. We demonstrate that the answer to this question is closely linked with the *efficiency* of transforming mate-pairs into mate-reads (the percentage of mate-pairs successfully transformed into mate-reads). When the read length exceeds a certain threshold, the *read length barrier*, the efficiency reaches nearly 100%, so that the read length indeed does not matter. For example, for the *E. coli* genome, the read length barrier is ≈ 35 nt.[1] This is good news for technologies with reads already longer than 35 nt (e.g., Illumina reads) but bad news for technologies with shorter reads (e.g., Helicos and SOLiD reads). However, while

[1] We emphasize that the read length barrier depends on the genome, the span of mate-pairs, coverage, error-rates in reads, variability in gap length, etc. The read length barrier $\approx 35nt$ for *E.coli* was computed under the assumption that the span is 300 ± 30 nt.

the current parameters of Illumina reads may be already sufficient for reliable assembly of some bacterial genomes, they are not sufficient for slightly larger genomes like *Saccharomyces cerevisiae* with higher read length barrier. This observation reveals a synergy between EULER-USR error-correction approach to increasing the read length and EULER-USR approach to transforming mate-pairs into mate-reads. Indeed, while the length of the "usable" portions of Illumina reads is currently below the read legth barrier for yeast (Figure 6.1), EULER-USR error correction allows one to increase the effective read-length beyond 45 nt thus exceeding the read length barrier. Therefore, while the mate-paired information represents by far the most important factor for improving the assembly quality, the read length also provides a valuable contribution to the assembly.

The EULER-USR software for assembling mate-paired short reads is publicly available at `http://euler-assembler.ucsd.edu`.

6.B Methods

We have developed the EULER-USR algorithm for assembling mate-paired and error-prone ultrashort reads. In addition to the previous Eulerian approaches [106] that correct reads based on k-mer multiplicities, EULER-USR corrects reads based on how they map to *repeat graphs* [106, 24], a generalization of the de Bruijn graphs.

The de Bruijn graph of a genome is constructed on the set of of all k-mers in the genome. Vertices u and v are connected by a directed edge (u, v) if there is a $(k+1)$-mer in the genome that has the k-mer u as a prefix and the k-mer v as a suffix. A small example of a de Bruijn graph is shown in Figures 6.2a,b. As a result, every substring of length $> k$ in the genome maps to a unique path in its de Bruijn graph. Similar to the de Bruijn graph of a genome (Figure 6.2c), one can construct the de Bruijn graph of reads (Figure 6.2d) on the set of all k-mers present in reads. The de Bruijn graphs of real genomes are very complex, a reflection of a large number of repeats with slightly varying repeat copies. A repeat graph of a genome (or reads) is a "simplified" version of the de Bruijn graph with small bulges and whirls removed [107, 91] (Figures 6.2e,f). The key observation in the Eulerian assembly is that the repeat graph of a genome can

be approximated by the repeat graph of reads and thus may be constructed from reads alone, i.e., without knowing the genome [107] (compare Figures 6.2e and 6.2f).

If the repeat graph of the genome is known, one may correct errors in a read by simply mapping this read to a path in the repeat graph and substituting the read by the path.[2] While this procedure would result in a nearly error-free set of reads, it is not clear how to construct a repeat graph from inaccurate reads and how to further map such reads to the repeat graph. In our approach, we construct the repeat graph from (accurate) read prefixes and then map entire reads (with inaccurate suffixes) to this graph by *threading*.

The EULER-USR algorithm consists of the following 3 steps that can be further supplemented by the threading procedure that we will describe later (section 6.B.4):

- Detecting accurate read prefixes and correcting errors within them using frequent k-mers (multiplicity m and higher). This operation generates the set of extremely accurate (nearly error-free) read prefixes.
- Constructing the repeat graph on error corrected prefixes using k-mers.
- Simplifying the repeat graph after transforming mate-pairs into mate-reads.

6.B.1 Detecting and error-correcting accurate read-prefixes

Although the quality of Illumina reads deteriorates with length, the prefixes are quite accurate (less than 2% error rate). While many reads can be turned into error-free reads by our error correction algorithm, errors will remain in low-quality reads even after error correction. In this case, it is important to detect the longest read prefix that may be error-corrected and discard the reads that cannot be corrected.

We correct reads using a modified version of the *Spectral Alignment (SA)* algorithm described in [106] and [23]. The *SA(Spectrum,read)* method corrects *read* given a set of k-mers *Spectrum*. Given a set of reads $\mathcal{R}$ and a frequency threshold m, we define a spectrum $Spectrum = Spectrum_k(\mathcal{R}, m)$ as the set of all k-mers that appear at least m times in reads from from $\mathcal{R}$. The set of such *solid* k-mers approximates the set of all k-mers in the genome. We define a read as *error-free* if all its k-mers are solid, otherwise,

[2] This procedure may also result in *error corruption* [106].

we attempt to make a read error-free by mutating a few nucleotides in the read [106, 24]. We only consider substitution mutations since there are few insertions/deletions in Illumina reads.

We use a greedy heuristic to find the minimum number of mutations to make every k-mer in a read solid. It records the number of k-mers that are made solid for all 3 possible mutations at every position in the read. The mutation that makes the highest number of k-mers in the read solid is applied if it "solidifies" more than t k-mers, where t is an internal threshold [24]. This heuristic is iteratively applied either until all k-mers in the read become solid, or until there are no mutations that can solidify more than t k-mers. We output the prefix of the read that is solid, or discard the read if none of it is solid. This partitions all reads that are not discarded into an (accurate) *prefix* and (less accurate) *suffix* that will be corrected at a later stage. The set of reads is divided into two partitions: *fixed* and *unfixable*. The fixed partition has reads that have a solid prefix, and the unfixable partition contains reads with no solid k-mers. A similar method is used in[20].

In order to choose the multiplicity threshold m, we assume that k-mers are generated from a mixture of two models: one erroneous, and the other correct (Figure 6.3). If we assume positional independence of errors in reads, the multiplicity of erroneous k-mers follows a Poisson distribution, and the multiplicity of correct k-mers follows a Poisson with a large mean that may be approximated by a Gaussian. We choose m by fitting a Poisson and Gaussian mixture model to the distribution of k-mer multiplicity and finding the first local minimum of this distribution (Figure 6.3).

6.B.2 Constructing the repeat graph on error-corrected read prefixes

We construct the de Bruijn graph on the error-corrected read prefixes and further transform it into the repeat graph. Reads that contain errors or SNPs in the middle create short undirected cycles in the de Bruijn graph called bulges, and reads that contain errors in the end create erroneous sources or sinks. Finally, some read errors form chimeric reads by transforming the sequence in one end of the read to that of a distant part of the genome, creating an edge that erroneously connects two unrelated contigs. Transformation of the de Bruijn graph into the repeat graph amounts to removing bulges,

erroneous sources/sink edges, and chimeric reads as described in [24].

In high-coverage projects, the partition of fixed reads created by error correction provides sufficient coverage across the genome to create a repeat graph that encodes the entire genome. However, many sequencing projects still contain low-coverage regions (often due to sampling bias). Since the error correction step relies upon redundancy of k-mers in reads, the reads in regions of low coverage may be discarded at this step, causing fragmentation of the assembled contigs. We found that the unfixable partition often contains many reads from the low-coverage regions, and their exclusion from the assembly causes fragmentation of the repeat graph.

In such cases it may be necessary to use what we call "Second Chance Assembly" – an assembly of all reads discarded during error correction (see Supplementary Material)

6.B.3 Simplifying the repeat graph by transforming mate-pairs into mate-reads

Since the initial proposal to use mate-paired reads for shotgun sequencing of the human genome [147], the mate-paired reads have been considered essential to *de novo* sequencing. Although in the past the short read mate-paired data have not been available, various next-generation sequencing vendors have recently released modules for high-throughput production of mate-pairs. EULER-USR utilizes the mate-paired reads by modifying the EULER-DB approach [104] for incorporating mate-pairs into the Eulerian assembly framework.

When mate-pairs are available, the input to the fragment assembly is a set of mate-pairs: "read$_{start}$ - GAP of length d - read$_{end}$". The key idea of EULER-DB [104] is using the repeat graph to transform the mate-pairs "read$_{start}$ - GAP of length d - read$_{end}$" into *mate-reads* "read$_{start}$ - SEQUENCE of length d - read$_{end}$" by filling in the gap of length d with an appropriate path in the repeat graph. As a result, EULER-DB has an ability to generate contiguous long reads of length $2 \cdot l + d$ from short mate-paired reads of length l. These long mate-reads are subjected to the traditional Eulerian assembly afterwards. In reality, the gap length is not fixed but varies from $d-\delta$ to $d+\delta$. Each mate-read corresponds to a *mate-path* between the mate-paired reads mapped to the repeat graph.

While EULER-DB worked well for traditional Sanger sequencing [107], it needs to be modified for short reads. The key parameter of EULER-DB is the *efficiency*, the percentage of mate-pairs successfully transformed into mate-reads. This transformation is trivial if there is a single path of length $\approx d$ between read_{start} and read_{end} in the repeat graph, as the path is simply used to fill the gap between the reads. This was indeed the case for the overwhelming majority of Sanger reads making EULER-DB rather efficient. However, the repeat graphs for short reads are often very complex and in some cases there are multiple paths of length $\approx d$ between paired reads (Figure 6.4). An example of a complex subgraph of the *E. coli* repeat graph that further illustrates this point is given in the Supplementary Material. Pevzner and Tang, 2001 [104] described this problem and proposed an iterative approach to solve it (see Figure 4a in [104]). EULER-SR [24], Velvet [151], and ALLPATHS [20] all implement the idea of filling the gap between mate-pairs using the repeat graph and apply it to simulated data (e.g., compare Figure 4b in [104] with Figure 5c in [20]). Below we apply EULER-DB to real Illumina mate-paired reads and show how to extend EULER-DB to analyze complex repeat graph characteristic for short read sequencing.

The "Breadcrumb" "All paths definition" procedures in Velvet and ALLPATHS are both aimed at filling up the gap between the mate-paired reads in the repeat graph. For most mate-pairs in *E.coli* there is only one path between mate-paired reads. In such cases the paths found by the "Breadcrumb" and "All paths definition" methods are equivalent to EULER-DB. However, the remaining mate-pairs are important for resolving complex tangles and Velvet and ALLPATHS describe different approaches to addressing this challenge. Below we describe how a simple extension of EULER-DB addresses this problem.

When there are multiple paths in the repeat graph between a mate-pair (read_{start}, read_{end}), we may choose the most likely path by measuring the *support* each path has from mate-pairs. Figure 6.4 illustrates the situation when there are many ways to transform the mate-pair (read_{start},read_{end}) into a mate-read using one of the paths between them. Let edge $e_{start}(e_{end})$ be the edge where read_{start} begins (read_{end} ends). A mate-pair supports a path P between e_{start} and e_{end} if one of the reads is in either e_{start} or e_{end}, and the other read is in an edge in P. The number of such mate-pairs

for a path P is denoted $support(e_{start},\ e_{end},\ P)$. The path P with the highest value of $support(e_{start},\ e_{end},\ P)$ is the most likely path, and is used to create the mate-path $P^{+} = (e_{start},\ P,\ e_{end})$. Note that due to gaps in coverage there may be edges in the most likely path that are not supported. Furthermore, errors in reads may create reads that support edges not on the optimal path. Therefore, we use a path with maximal $Support(P)$ among all paths between $read_{start}$ and $read_{end}$ unless $Support(P)$ falls below $MinSupport$ threshold. Further details of transforming mate-pairs and into mate-reads are given in the Supplementary Information.

6.B.4 Assembling error-prone reads (error-correction by threading)

Each read corresponds to a unique *read path* in the de Bruijn graph representing the sequence of the read. Since the repeat graph approximates the de Bruijn graph, a similar argument applies to the read-paths in the repeat graph. A read may be mapped to the de Bruijn graph by aligning it to a closest subpath in the graph. Since the de Bruijn graph is built on read prefixes, the path corresponding to every read prefix (*prefix-path*) is known thus facilitating the read mapping. We may find the path that the entire read maps to by searching all subpaths continuing from the known prefix-path, a process we refer to as *threading reads through a graph*, or simply *threading*.

In the case that threading is invoked, the EULER-USR(k, m, l) algorithm proceeds in five steps. The user has a choice of either specifying all three parameters: k, m, and l, or specifying a single parameter k. In the latter case, EULER-USR selects the suitable parameters m and l automatically.

- Detecting accurate read prefixes and correcting errors within them using frequent k-mers (multiplicity m and higher). This operation generates the set of extremely accurate (nearly error-free) read prefixes.
- Constructing the repeat graph on error corrected prefixes using k-mers. This operation generates the set of *k-mer contigs*.
- Threading entire reads through the repeat graph to extend the effective read length. This operation generates the set of accurate *threaded* reads.

- Constructing the repeat graph on threaded reads and generating l-mer contigs using l-mers ($l > k$).
- Simplifying the repeat graph by transforming mate-pairs into mate-reads

Once the repeat graph has been constructed on the (accurate) read prefixes, we attempt to map every fixed read to the graph. However, while mapping of the (accurate) read prefixes is well defined, mapping of (inaccurate) read suffixes is ambiguous. EULER-USR utilizes the repeat graph to correct errors in read suffixes.[3]

Every read *prefix* $*$ *suffix* not discarded by error correction is represented as a concatenation of its *prefix* and *suffix*. Since the genome is an Eulerian traversal of its repeat graph, all substrings of the genome map to paths in the repeat graph. While the accurate *prefix* may be uniquely mapped to a path $path(prefix)$ in the repeat graph, it is not clear how to map the entire read *prefix* $*$ *suffix* since *suffix* is inaccurate. We argue that to map *prefix* $*$ *suffix*, one has to choose one of the extensions of $path(prefix)$ among all paths of length n (read length) that begin with $path(prefix)$. We denote the set of such paths as $\mathcal{P}$ and argue that a path in $\mathcal{P}$ with the minimum edit distance to the read represents the "best" mapping of the read *prefix* $*$ *suffix* to the repeat graph. While in many cases such a *thread-path* $path(prefix * suffix)$ may be used to correct the read *prefix* $*$ *suffix*, it has to be done with caution (see below).

If $\mathcal{P}$ has a single edge (Figure 6.5a), we correct the read with the sequence on the edge. However, these reads may not be used to resolve repeats (because they are on a single edge), nor improve consensus quality (the consensus is used to fix the read), thus, low quality reads that map to a single-edge are vestigial in terms of improving the assembly.

If $\mathcal{P}$ has multiple paths (Figure 6.5b), we rank paths $P_1, P_2, \ldots$ in $\mathcal{P}$ in the increasing order of their Hamming distances $dist(P_1), dist(P_2), \ldots$ to the read *prefix* $*$ *suffix*. While it is tempting to choose the "optimal" path P_1 for correcting errors in the read *prefix* $*$ *suffix* it has to be done with caution. The problem is that the sequencing errors in the inaccurate *suffix* may transform it into an alternative string that maps to a "wrong" path in the repeat graph. We therefore check that (i) the optimal path P_1

[3]Error-corrected read suffixes only contribute to enlarging the assembled contigs and do not contribute to basecalling.

is sufficiently similar to *prefix* $*$ *suffix*, and (ii) the second best path P_2 is sufficiently dissimilar from *prefix* $*$ *suffix*. To check these conditions, we use a parameter f, the expected error rate in read suffixes, and classify a path P as similar to the read if $dist(P) \leq f \cdot |suffix|$, and dissimilar otherwise. If both conditions (i) and (ii) are satisfied we use P_1 for correcting the read *prefix* $*$ *suffix*, otherwise we iteratively trim before the position of the last difference, and re-thread the read until both these conditions are satisfied.

To obtain the final assembly that takes advantage of the longer threaded reads, we build the repeat graph on l-mers for $l > k$ (where k was used to build the original repeat graph) so that repeats of length l and shorter are resolved. The tradeoff between the k-mer size and repeat resolution is that edges from the repeat graph G of the genome (constructed on k-mers) will be split in the repeat graph of reads if there is a gap longer than $n - k$ between read start positions, where n is the read length. When $l > k$ the repeat graph of reads constructed on l-mers will be more fragmented than the repeat of reads constructed on k-mers. We prevent fragmentation of the repeat graph of reads constructed on l-mers by creating artificial reads $x \circ y$ for every pair of adjacent edges (contigs) x and y in the repeat graph of reads constructed on k-mers (k-mer contigs). One can either set the maximal $l = n-2$ or empirically chose l to minimize fragmentation of the graph.

6.C Results

6.C.1 Datasets

We use the following datasets to benchmark EULER-USR and compare it with Velvet [151], the most accurate among the recently published short read assemblers.

- ECOLI. A set of 29.8 million paired Illumina reads from the $\approx$ 4.6 Mb *E.coli* genome (227X coverage). For benchmarking we randomly selected 10 million reads from this dataset. Mapping of reads to *E.coli* genome revealed that 87% are error-free. The separation between mate-pairs is 145 ± 25 bp.

- BAC50 and BAC35. A set of Illumina reads from an $\approx 170Kb$ human BAC gen-

erated at the Joseph Ecker lab at the Salk Institute.[4] This BAC was thoroughly sequenced over several runs allowing us to generate an error profile that was not biased to a single run. A total of 2 million 35-50 base reads were generated for this BAC resulting in 500X coverage allowing us to choose an appropriate subset for a typical coverage benchmark. We randomly selected reads resulting in 50X coverage by 50 nt long reads. Mapping these reads from the BAC to the reference sequence revealed that 20% are error-free and 12% have only 1 error. Furthermore, 84% are error free in the first 30 bases.

While EULER-USR is designed to work with longer reads, the Velvet assembler is optimized for 35 base long reads and the performance deteriorates for longer reads. In order to make a fair comparison with Velvet we created a 35 nt read dataset BAC35 by truncating 50 nt reads from BAC50 to 35 nucleotides resulting in 35X coverage of the BAC.

- simBAC100 and simECOLI100. A set of simulated 100 base reads from the human BAC. This set was generated to check whether EULER-USR can support extending sequencing reactions well beyond their prime. We simulated reads by mapping all 2 million reads from the dataset generated at Salk Institute to the BAC, extending them up to 100 bases, and simulating random errors. We further selected a set of resulting 100 base reads so that the coverage was 200X (to ensure that results were not biased by gaps in coverage). Errors in the resulting 100 base long reads were simulated using a 1% error rate for the first 35 bases, and 20% error rate for the remaining 65 bases. This error profile leads to a challenging assembly without making many assumptions about the distribution of errors along the reads. Our method for correcting errors using a repeat graph requires that the entire genome is covered by the high quality prefixes of reads. The simBAC100 dataset includes as many reads as the BAC50 dataset to ensure that the entire genome is represented in the repeat graph constructed on 35 bases read prefixes.

  The simECOLI100 is the set of simulated 100 base reads from *E. coli* (100X coverage). Errors were added to the reads according to the same 1%-20% error profile

[4] This BAC has a repeat content representative of the rest of the human genome.

used for simBAC100.

- simBAC35. A set of 35 base read prefixes from simBAC100. Because the simBAC100 dataset uses simulated reads, it is not directly comparable to the assemblies on real reads. Instead, to perform proper comparison for the effect of threading reads, we compare the assembly on 100 base (inaccurate) simulated reads to the assembly on the 35 base (accurate) simulated reads.

6.C.2 Benchmarking

We compared five recently published *de novo* short read assemblers aimed at short reads: SSAKE [146], SHARCGS [35], VCAKE [63], Edena [53], and Velvet [151]. Of all assemblers, Velvet performed the best in terms of N50 contig size.

SSAKE is designed for assembling error-free reads, and SHARCGS is designed for reads with error-rate below 0.05%. Running these two methods on Illumina reads resulted in filtering of all reads on a preprocessing step or (if preprocessing is turned off) in an inferior assembly quality. VCAKE is an improvement of SSAKE and is able to assemble reads with higher error rate. For the BAC35 dataset, VCAKE, Velvet, Edena, and EULER-USR produced similar assemblies. However, the quality of assemblies generated by VCAKE or Edena deteriorate with increasing the read length (BAC50 dataset) producing more fragmented assemblies than Velvet. Furthermore, the assemblies for the ECOLI dataset by both Velvet and EULER-USR resulted in doubling the N50 contig size as compared to VCAKE or Edena. We therefore decided to limit the detailed benchmarking to Velvet only.

A goal in the Eulerian approach is to construct the repeat graph $G_k(Reads)$ on the set of *Reads* that best approximates the "ideal" repeat graph $G_k(Genome)$ of the *Genome* [24] (denoted as REPEAT-GRAPH(k) in the follow-up Tables and Figures), as every Eulerian path in the repeat graph corresponds to a possible solution of the fragment assembly problem. However, due to fragmentation, the REPEAT-GRAPH statistics are misleading because the longer the contigs are, the more likely they are to be fragmented by low coverage regions (note that fragmentation of long contigs leads to a quick deterioration of the N50 size). Therefore, uneven distribution of reads over the genome may turn approximating $G_k(Genome)$ into an unattainable goal. To setup a

more realistic goal, we transform the set *Reads* into the error-free set *PerfectReads* (by substituting every read with the sequence of the genome it maps to). In the absence of mate-pairs, the repeat graph $G_k(PerfectReads)$ constructed on this set of reads represents the best assembly our assembler aims for while assembling the real reads (refereed to as OPTIMAL-ASSEMBLY in the follow-up Tables and Figures).

6.C.3 How are assemblies improved by mate-paired reads?

To benchmark EULER-USR and VELVET on the ECOLI dataset, we first evaluated assembly with unpaired reads in order to later gauge the effect of mate-pairs. Both the EULER-USR and Velvet ($k = 27$) assemblies were close to the theoretically optimal assembly (Table 6.1 and Figure 6.6) with similar N50 sizes (20K for EULER-USR and 16K for Velvet) and no miss-assemblies. The contigs longer than 500 bases in EULER-USR assembly (those likely to be non-repetitive) contain 6 mismatches, 3 insertions, and 1 deletion (30 mismatches, 3 insertions, and 3 deletions in the Velvet assembly). This analysis underestimates the basecalling errors by limiting them to long contigs and thus avoiding the most difficult repeated regions. Nevertheless, the basecalling accuracy appears to be comparable or even better than the accuracy of high-coverage Sanger sequencing.

Since most mate-pairs are separated by $\approx 145 \pm 25$ nt, the average length of transformed mate-reads for this dataset is $2 \cdot 35 + 145 = 215$ bp. Table 6.1 and Figure 6.6 compare EULER-USR and Velvet and illustrate that mate-pairs significantly improve the assemblies. The N50 length for *E.coli* assembly increases from 16K to 45K for Velvet and from 19K to 62K for EULER-USR. EULER-USR generated 127 contigs longer than 1000 bp that is comparable to the typical number of contigs resulting from pre-finished Sanger assembly of bacterial genomes (see [107] and http://nbcr.sdsc.edu/euler/benchmarking/bact.html). An alignment of the assemblies to the *E. coli* genome is shown in the Supplementary Materials.

6.C.4 How are assemblies improved by read-threading?

When estimating input parameters for EULER-USR, our mixture model suggested the multiplicity threshold $m = 5$ for BAC50 dataset with k-mer size 20. The

Table: 6.1 A comparison of assemblies of *E. coli* 35 base Illumina reads, unpaired and mate-paired. *N50*: the size of the contig such that 50% of the assembly is contained in contigs of size N50 or greater. *Length (# contigs) (>20000 nt)*, total length of all contigs longer than 20000 and total number of such contigs. We found that Velvet produces optimal results when run as Velvet(27,5). Missassemblies were counted as contigs that have portions out of order that are greatr than 150 bases in length, to account for repeat consensus basecalling.

Assembly	N50	Len. (# contigs) >20000 nt	Len. (# contigs) >5000 nt
REPEAT-GRAPH(30)	22,173	2,432,772 (69)	4,232,578 (237)
EULER-USR unpaired	20,096	2,233,252 (68)	4,212,353 (249)
Velvet unpaired	16,424	1,953,255 (59)	4,068,326 (262)
EULER-USR mate-pairs	62,015	4,207,753 (72)	4,481,764 (96)
Velvet mate pairs	45,427	3,800,552 (79)	4,419,542 (131)

Assembly	Len. (#contigs) >1000 nt	# miss assemblies
REPEAT-GRAPH(30)	4,484,685 (331)	-
EULER-USR unpaired	4,490,810 (355)	0
Velvet unpaired	4,484,065 (416)	0
EULER-USR mate-pairs	4,524,074 (113)	2
Velvet mate pairs	4,507,932 (167)	0

accuracy of error correction was evaluated by mapping error-corrected reads to the BAC (Table 6.2). ¿From the original sets of reads, 91.3% of the BAC35, 88.6% of the BAC50 datasets, and 98.0% of the simBAC100 dataset were retained after error-correction based on Spectral Alignment. During graph correction, the removal of certain edges from the graph truncates the reads that map to these edges, further shortening the average read length. While most reads in this dataset were trimmed only by a few nucleotides, others become rather short and need to be extended with threading as described above.

Table: 6.2 Error rate (per read base) and average length of reads on different stages of the EULER-USR threading algorithm. The error rate is computed by mapping reads to the genome. We compute the length and error rate for the original reads, reads corrected by Spectral Alignment that are retained after graph correction, and finally after threading. The increased error rate after threading is due to threading reads in a repeat through their consensus sequence in the repeat graph.

	Original reads		SA corrected reads			Threaded reads after graph corr.	
Dataset	Len.	Error rate(%)	Avg. length	Error rate(%)	Retained reads (%)	Avg. length	Error rate(%)
BAC35	35	0.92	34.9	0.01	91.3	34.9	0.004
BAC50	50	4.36	46.7	0.04	88.6	49.3	0.049
simBAC100	100	13.3	46.6	0.07	98.0	94.5	0.050
simECOLI100	100	12.6	50.5	0.003	99.6	98.8	0.017

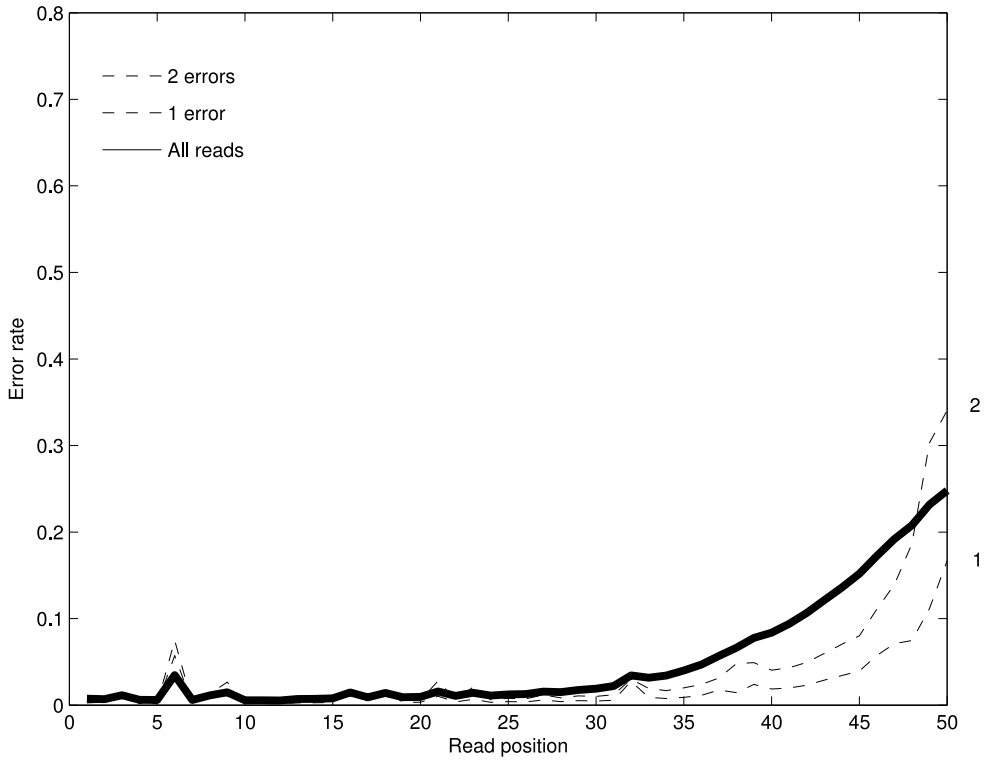


Figure 6.1 The positional profile of basecalling errors for Illumina reads for 2 million 50 nt long reads from a human BAC. The error rate across reads is shown (solid) along with the error rate for reads with a fixed number of errors. The erroneous nucleotides in each read are detected by mapping the read to the reference genome. The high error rate in position 6 is due to the bias in our particular data set rather than a systematic problem with the Illumina technology.

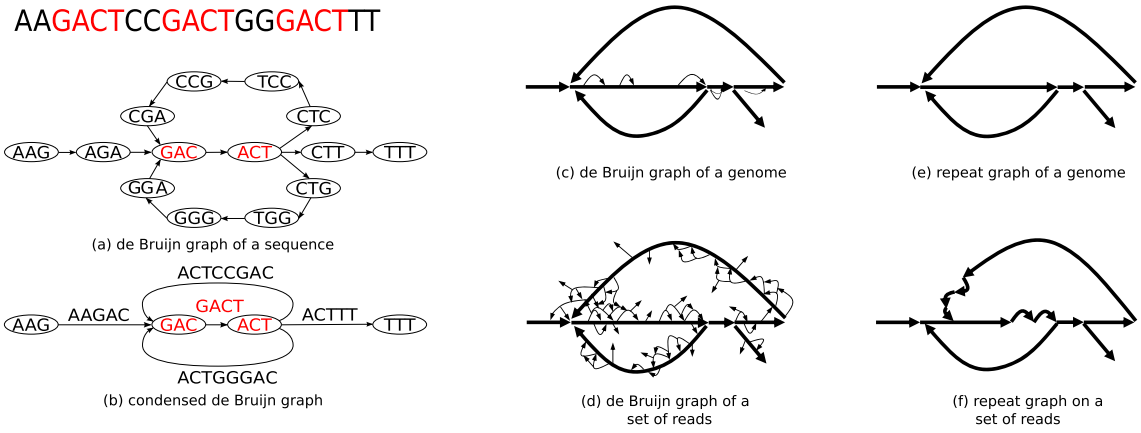


Figure 6.2 From de Bruijn graphs to repeat graphs. The de Bruijn graph of a sequence contains a vertex for every k-mer in the sequence, and an edge (u, v) for every pair of consecutive (overlapping) k-mers in the sequence (a). The condensed de Bruijn graph replaces all paths containing non-branching vertices by a single edge labeled by the sequence that generated the path (b). When the condensed de Bruijn graph is constructed on a genome, it contains some small bulges and whirls representing repeats with slightly varying repeat copies (c). In the repeat graph the bulges and whirls are removed (e). The de Bruijn graph of reads contains additional spurious bulges and whirls caused by sequencing errors in reads (d). The goal of the Eulerian assembly is to construct the repeat graph of reads (f) that approximates the repeat graph of the genome. Different papers use different terminology, e.g., the edges of these graphs are referred to as "blocks" in [151] and "unipaths" in [20].

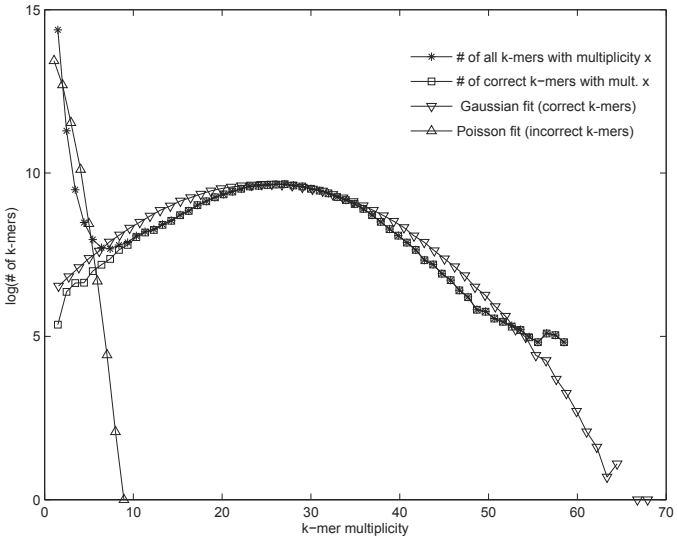
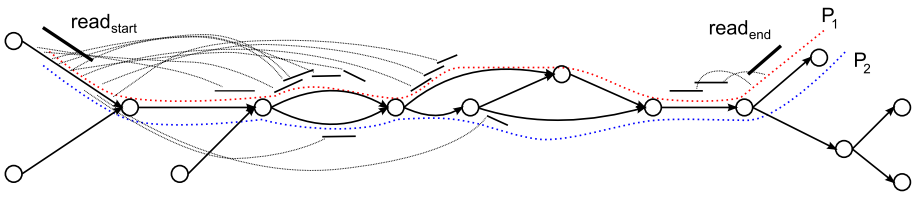
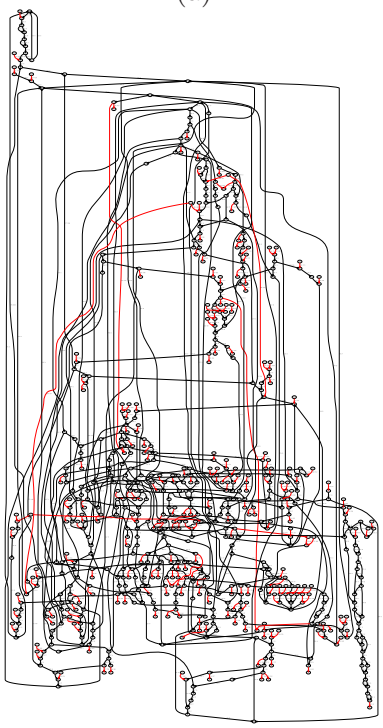


Figure 6.3 Choosing the multiplicity threshold for error correction. All k-mers appearing in the reads are classified as *correct* if they appear in the genome and *incorrect* otherwise. For a multiplicity x, let $correct(x)/incorrect(x)$ be the number of correct/incorrect k-mers with multiplicity x (the plots are shown for 50 base long Illumina reads from a human BAC and $k = 20$). As expected, most high-multiplicity k-mers are correct and most low-multiplicity k-mers are incorrect. A Poisson/Gaussian mixture model was fit to the distribution of all k-mer multiplicities in order to model the process of generating incorrect (Poisson) and correct k-mers (Gaussian). To show the fit of the model, the k-mer multiplicities were generated according to the estimated parameters λ=0.95, μ=25, and σ=9.38, with a mixing parameter $\omega = 0.95$. One may find the multiplicity m with good separation between correct and incorrect k-mers by estimating the first local minimum from the distribution of k-mer counts, or the minimum of the sum of the probabilities of the mixture model, a more smooth distribution. For multiplicity threshold m=5, only 0.6% of correct 20-mers have multiplicity <5 and only 0.3% of incorrect 20-mers have multiplicity $\geq$5.



(a)



(b)

Figure 6.4 (a) A fragment of a made-up repeat graph formed by three divergent copies of a repeat. There are many possible paths from read$_{start}$ to read$_{end}$. To transform the mate-pair "read$_{start}$ - GAP of length d - read$_{end}$" into a mate-read "read$_{start}$ - SEQUENCE of length d - read$_{end}$", we compute the support for every path between read$_{start}$ and read$_{end}$ and select a path with maximum support. In this example, the "red" path P_1 has greater support than the "blue" path P_2. (b) A fragment of the real repeat graph *E. coli* containing one of the most common repeats. This demonstrates the complexity of the portion of the graph which is searched in order to define mate-reads. Red edges are long edges that are unique in the genome, from which reads read$_{start}$ and read$_{end}$ are considered. Black edges are repeats that are gapped by a mate-pair, and resolved using support from mate-pairs in read$_{start}$ and read$_{end}$.

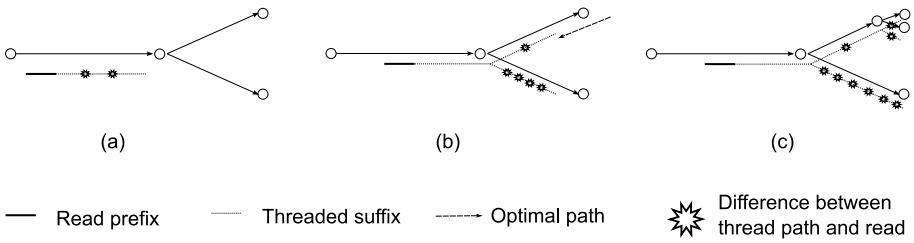


Figure 6.5 Mapping reads to the paths in the repeat graph. In (a) a read maps to a single edge. In (b) a read maps to two paths, and the closest one is chosen. In (c) a read may be mapped to two similar paths implying that trimming is required.

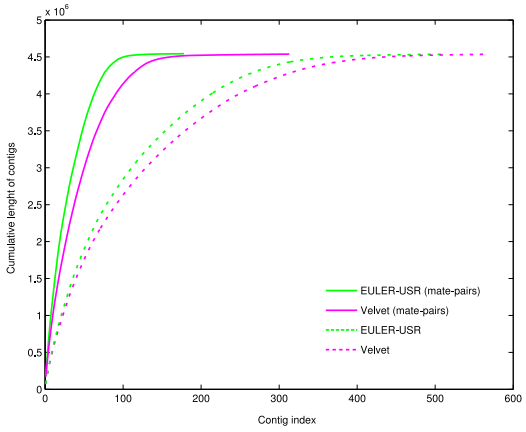


Figure 6.6 Comparison of EULER-USR and Velvet using both paired reads, and the same reads with mate-pair information removed (ECOLI dataset). The contigs are ordered in the decreasing order of sizes and the cumulative size of x longest contigs is shown on the y-axis.

Table: 6.3 Assembly statistics of various datasets of Illumina reads. *N50*: the size of the contig such that 50% of the assembly is contained in contigs of size N50 or greater. *Length (>1000)*, *Length (>500)*, and *Length (>100)*: the total length of all contigs longer than 1000, 500 nt and 100 nt, respectively. For Velvet($k-mer\ size, coverage$) we found that the coverage cutoff t=5 maximizes the assembly quality. The effect of threading reads on assembly quality may be seen by comparing simBAC35 and simBAC100. The rows describing REPEAT-GRAPH(50) and OPTIMAL-ASSEMBLY(50) are identical since the reads cover the entire BAC in simBAC100 dataset.

Assembly method/dataset	N50	Len. (# contigs) >1000	Len. (# contigs) >500	Len. (#contigs) >100	# miss assemblies
BAC35					
REPEAT-GRAPH(25)	1869	97946 (34)	126774 (74)	153378 (194)	-
OPTIMAL-ASSEMBLY(25)	1609	92758 (39)	119773 (78)	153348 (225)	0
EULER-USR(20,2,25)	1786	89905 (39)	118576 (80)	147227 (210)	0
Velvet(21,5)	1428	87551 (43)	113522 (80)	138554 (173)	0
BAC50					
REPEAT-GRAPH(40)	4168	143224 (39)	160679 (64)	175246 (128)	-
OPTIMAL-ASSEMBLY(40)	2023	129392 (55)	152551 (87)	176491 (193)	0
EULER-USR(20,5,40)	2022	119489 (45)	148319 (86)	171082 (164)	0
Velvet(31,5)	1381	84292 (39)	112886 (78)	139176 (169)	0
simBAC35					
REPEAT-GRAPH(25)	1869	97946 (34)	126774 (74)	153378 (194)	-
OPTIMAL-ASSEMBLY(25)	1847	95180 (35)	159359 (85)	175305 (242)	0
EULER-USR(20,5,25)	1818	98187 (39)	129671 (85)	162109 (242)	0
Velvet(21,5)	1844	97174 (36)	122816 (73)	144578 (153)	0

Assembly method/dataset	N50	Len. (# contigs) >1000	Len. (# contigs) >500	Len. (#contigs) >100	# miss assemblies
simBAC100					
REPEAT-GRAPH(50)	7163	167112 (31)	172990 (39)	175444 (53)	-
OPTIMAL-ASSEMBLY(50)	3971	162129 (47)	169794 (58)	176908 (94)	0
EULER-USR(20,5,50)	2639	140718 (47)	163244 (80)	175900 (135)	0
Velvet(31,5)	695	36796 (22)	77557 (80)	120147 (241)	0
simECOLI100					
REPEAT-GRAPH(50)	59656	4519592 (140)	4528404 (152)	4583803 (533)	-
EULER-USR(20,10,50)	44710	4519083 (182)	4531837 (200)	4562551 (366)	1

Table 6.3 presents the statistics of of the N50 contig size as well as the cumulative contig size for various datasets (reported for contigs longer than 1000, 500, and 100 bases). Since the N50 statistic is limited we show the differences in assemblies by plotting the cumulative length of contigs ordered by size (Figure 6.7). Figure 6.7 shows how longer threaded reads improve the assembly quality for both real and simulated reads.[5] Figure 6.7 illustrates that while VELVET and EULER-USR show similar results for the BAC35 dataset (no read threading), the EULER-USR assembly improves for BAC50 and simBAC100 datasets (due to its ability to utilize the error-prone reads) while VELVET assembly remains the same. Indeed, even with a modest increases in read length from 35 nt to 50 nt, read threading increases N50 contig size by 13% and the total length of long contigs (¿1000 nt) by 22%. Figure 6.8 shows how assembly improves with increase in read coverage when assembling the *E. coli* genome and leads to a conclusion that the coverage increase beyond 55X results only in modest increase in assembly quality.

When the BAC50 dataset is assembled without threading reads, the N50 contig size is 1752 with a net assembly size of 171301. Read-threading only improves the quality of assembly by correcting reads that pass through 3 or more edges (most reads map to one or two edges in the repeat graph). Table 6.4 shows the number of reads that are correctly fixed with threading and how many edges they are threaded through. Although a very small fraction of reads are threaded through more than 3 edges, they improve the quality of assembly.[6]

[5] In some cases the statistics for EULER-USR is slightly better than for OPTIMAL-ASSEMBLY due to subtle differences in contig reporting.

[6] For BAC50 dataset, EULER-USR has 20 mismatches and 2 insertions, a higher error rate as compared to ECOLI dataset. The Velvet assembly on this dataset contains fewer errors: 4 mismatches and 7 deletions.

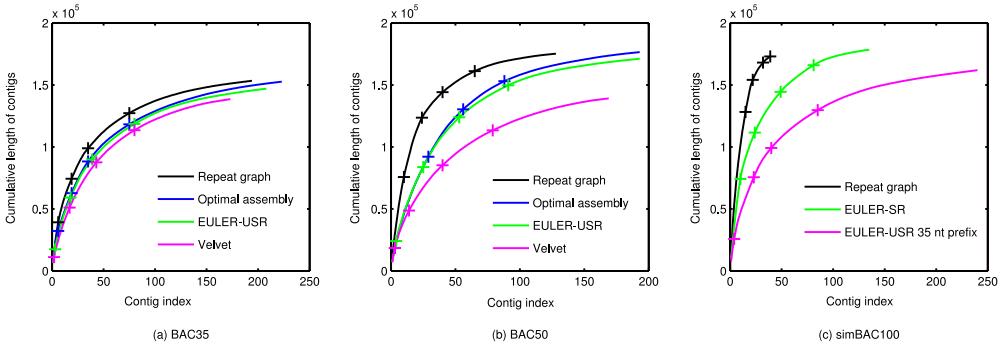


Figure 6.7 Comparison of EULER-USR (threading) and Velvet. In each plot, the contigs are ordered in the decreasing order of sizes and the cumulative size of x longest contigs is shown on the y-axis (only contigs longer than 100 bases are shown). See Table 6.3 for the choice of parameters of all programs in these plots. For all assemblies of the BAC, the locations of the contigs closest to lengths 5000, 2000, 1000, and 500 bases are shown with a "+" mark.

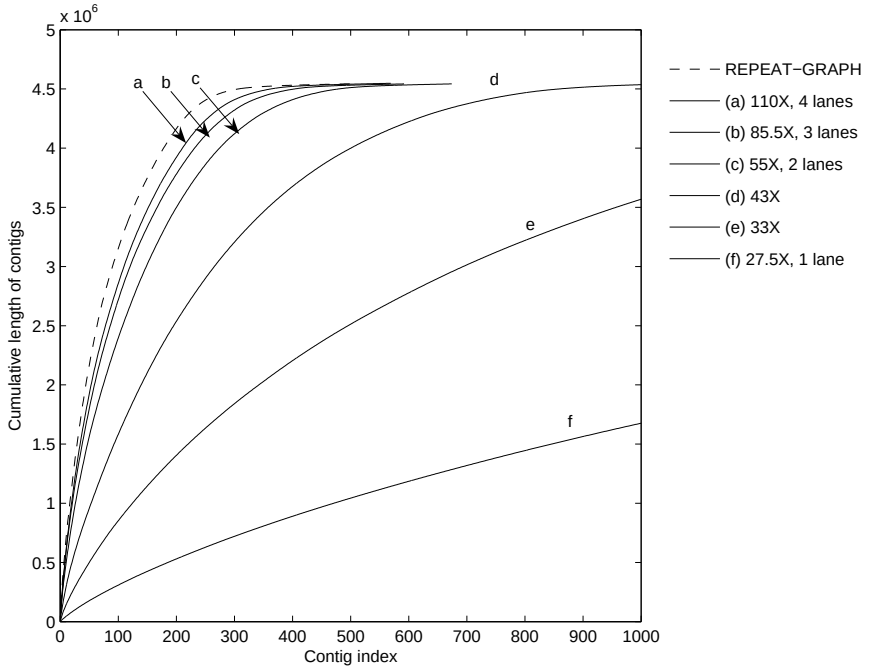


Figure 6.8 Statistics of assembly for various read coverages (*E. coli* genome). The cumulative length of contigs in order of decreasing length is shown for the 1000 longest contigs. The cumulative length of contigs of the repeat graph on the genome is shown as a dashed line.

Table: 6.4 The results of read threading. Reads are classified into four categories: correct/correct (if threading does not change a correct read), correct/incorrect (if threading turns a correct read into incorrect), incorrect/correct (if threading turns an incorrect read into correct), and incorrect/incorrect (if threading turns an incorrect read into an incorrect read). The table classifies reads in each of these four categories depending on how many edges in the repeat graph they span.

No. reads	Total	Reads spanning one edge	Reads spanning two edges	Reads spanning > 2 edges	Average read length (after threading)
BAC50					
correct/correct	100942	95781	2161	2550	50
correct/incorrect	207	174	27	6	50
incorrect/correct	55515	52408	1464	1643	42
incorrect/incorrect	347	254	33	60	45
simBAC100					
correct/correct	2819	2523	115	181	97
correct/incorrect	5	2	3	0	100
incorrect/corrected	323589	272290	16480	34819	96
incorrect/incorrect	4116	717	888	2511	91

Table: 6.5 Mapping of the 4136 genes from *E.coli* into theoretical repeat graph and repeat graph constructed by EULER-USR and Velvet for the ECOLI dataset (see Table 6.1 for a description of the parameters). We show the number of genes that are contained entirely within the constructed contigs.

Method	REPEAT-GRAPH	EULER-USR	Velvet
# complete genes	3937 (95.2%)	3956 (95.7%)	3912 (94.6%)

We analyzed the simBAC100 dataset to evaluate how threading improves the assembly quality. The Spectral Alignment error correction routine trims reads to on average 46.6 bases. However, after threading, the average read length was recovered to 94.5 bases. The longer read length allowed us to perform repeat resolution with a larger k-mer size (50), resulting in a further 18% improvement in N50 assembly as compared to simBAC35. There is a large disparity between the assembly on perfect reads of length 100, and the assembly on threaded reads, even though the average threaded read length is over 90 nt. Most of the repeats in the genome are due to repetitive elements longer than 100 bases. When the repeats diverge from the consensus sequence they may be resolved by perfect reads. However, we thread reads through the repeat graph, a simplified representation of the genome that merges repeats into a consensus. When the repeat consensus is longer than the average read length, small mutations that differentiate repeat copies are lost, and therefore are not resolved by reads only covering a portion of the repeat.

Since bacterial genomes have a compact gene structure, we analyzed how many genes are captured within contigs constructed by various programs. Even though the repeat graph of *E. coli* is fragmented into many contigs (over 500 for REPEAT-GRAPH(30)), many contigs are long, and contain multiple genes. We mapped the contigs of REPEAT-GRAPH, EULER-USR, and Velvet assemblies to the genome, and counted the number of genes that contained entirely within a contig. Table 6.5 illustrates that contigs captured by Velvet and EULER-USR capture a large number of bacterial genes thus enabling various applications. For example, one can perform MS/MS proteomics analysis of bacterial genomes with Illumina contigs almost as efficiently as with completed genomes [49].

The *E. coli* genome contains relatively few repetitive elements compared to the human genome, and so the longer reads should be able to resolve more repeats in

E. coli than in the human BAC. In the simECOLI100 dataset, the usable read length was increased from an average of 50.5 nt to 98.8 nt (Table 6.2) by threading. When we compare the assembly on the simulated read prefixes to the threaded reads, the N50 contig size more than doubles to ≈ 45K. This indicates that the announced increase in Illumina read length to 100 nt (planned for 2009) will lead to significant improvement in assembly in the case of unpaired reads. While it is an important improvement in applications like single cell sequencing (where the mate-paired protocols are not available yet), it remains unclear whether the read length (as opposed to span) matters in case of *mate-paired* reads (i.e., would increasing the length of mate-paired reads from 35 nt to 100 nt lead to significant improvements in assembly if the span remains fixed?)

6.C.5 Does the read length matter?

The availability of two methods to resolve repeats (mate-pairs and threading) brings the question whether they may be used in conjunction to further improve assemblies. To test this, we simulated mate-pairs with span 300±30 nt in the genomes of *E. coli* and *S. cerevisiae*. The read length r was fixed in each simulated dataset, with a minimum read length of 25 nt and maximum length 100 nt. The goal was to evaluate whether the quality of assembly (e.g., $N50(r)$, N50 length for reads of length r) with longer reads improves as compared to the quality of assembly with shorter reads (e.g., whether 100 nt mate-paired reads result in a better assembly than 35 nt reads). We define the *read length barrier* as the read length after which the quality of assembly does not significantly improve (e.g., N50 does not increase by more than 5%).

The error-free mate-paired reads were simulated starting at every genomic position and each simulated dataset with reads of length r was assembled with EULER-USR (k-mer size 24). To evaluate the quality of assemblies, we used N50 contig size and *efficiency* (the percentage of mate-pairs transformed into mate-reads). The efficiency should be analyzed with caution because only mate-pairs spanning multiple edges in the repeat graph contribute to improving the assembly. In the *E. coli* assembly of Illumina reads, this was only 3.9% of all mate-pairs. As a result, for the simulated *E. coli* assembly, the efficiency is rather high for all read lengths (varying from 97.8% for $r = 25$ to 99.0% for $r = 55$ and to 99.2% for $r = 100$). However, even a small increase in efficiency translates

into significant increase in N50 statistics (varying from $\approx$ 40K for $r = 25$ to $\approx$ 60K for $r = 55$). Therefore, small increases in efficiency may reflect very significant increase in the number of "useful" mate-pairs, i.e., mate-pairs that improve the assembly. To better gauge the contribution of such "useful" mate-pairs, we introduce the *relative efficiency* parameter, (the percentage of *useful* mate-pairs transformed into mate-reads (we call a mate-pair useful if its reads reside on different edges). For the yeast dataset, the relative efficiency varies from from 61% for $r = 25$ to 80% for $r = 55$ (the maximum value among all simulations is 81%). These results indicate that for the yeast dataset efficiency hardly change after the read length exceeds $\approx$ 60 nt.

Our attempt to answer the question "Does the read length matter?" is limited in many aspects (e.g., reads were simulated error-free, and coverage was perfectly uniform) and it only answers the question whether the read length matters for EULER-USR assemblies (rather than for a theoretically optimal assembly with mate-pairs).[7] However, it reveals that for the *E.coli*, the assembly hardly improves after the read length exceeds 35 nt (efficiency=98.7%, N50=60Kb). However, the assembly deteriorates when the read length decreases from 35 to 25 indicating that the read length barrier for *E.coli* (with the chosen simulation parameters) is $\approx$ 35 nt.[8]

For the *S. cerevisiae* genome, the assembly quality only slightly improves after the read length exceeds 60 nt (N50 is $\approx$ 70K for $r = 60$ but drops to *approx*62K at $r = 45$ and to $\approx$ 41K at $r = 25$. It indicates that the read length barrier for *S. cerevisiae* (with chosen simulation parameters) is $\approx$ 60 nt.

6.D Discussion

The recent addition of mate-paired reads to the arsenal of short read technologies opened a possibility of assembling complex genomes for a fraction of the cost of the traditional Sanger sequencing. We demonstrated that the Eulerian approach is well suited for assembling mate-paired short reads by transforming mate-pairs into mate-

[7] The theoretically optimal algorithms for assembling mate-paired reads remain unknown even for error-free reads and fixed distance between between mate-pairs [82].

[8] We found that assemblies of mate-pairs with average span $d \pm \sigma$ may be sensitive to the parameter σ even for the same d. For example, simulated assemblies with error-free reads may have lower quality than the real assemblies with the same d but different σ.

reads using repeat graph. We further complemented the approach from [104] by selecting the most "supported" mate-reads to resolve some difficult cases when a mate-pair may be transformed into multiple mate-reads.

In addition to incorporating mate-pairs into fragment assembly, we also show that the conventional wisdom of "read trimming" may be substituted by threading to correct error-prone read tails. We demonstrate that if a sequencing technique "suffers" from quality degradation along the length of a read, it may still be used effectively in *de novo* assembly. Despite the fact that short read assemblies are rather fragmented, we demonstrate that most bacterial genes map to single contigs thus enabling gene discovery and annotation of bacterial genomes.

The Eulerian approach models the error-prone suffixes of the reads as short edges to vertices of out-degree zero. All recently developed short read assemblers remove such edges from the graph (e.g., via the the *erosion* procedure in [107]) thus essentially discarding information contained in the error-prone read suffixes. Therefore, even if the reads are not explicitly trimmed, they are implicitly trimmed after the de Bruijn graph is constructed (e.g., using the "clipping" procedure in Velvet [151] or "removal of hanging ends" procedure in ALLPATHS [20]). EULER-USR differs from these approaches by utilizing information in the error-prone read prefixes.

Our study on the use of mate-paired reads in conjunction with read threading revealed that there exist some synergy between these two approaches when the read length remains below the read length barrier. While mate-pairs represent the major factor in improving the assembly quality, read threading contributes to further improvements in assembly. The next challenge for short-read technologies is to assemble larger and more complex genomes. The ability to exploit any information possible to resolve repeats will become important when assemblers move to mammalian genomes.

6.E Acknowledgments

This research is supported NIH grant 1R21HG004130-01 as well by NSF grant EIA-0303622. We thank Dirk Evers, Klaus Maisinger, and Jacques Retief for insightful discussions about the Illumina technology and to Xiaohua Huang and Eric Roller for

many discussions on emerging next-generation sequencing technologies. We are grateful to Ronan O'Malley and Joseph Ecker for providing us with the BAC reads.

6.F Second chance assembly

In low-coverage regions it may be necessary to use "Second Chance Assembly" to complement the standard EULER-USR assembly pipeline. The "Second Chance Assembly" attempts to assemble reads discarded during error correction along with 'hooks': edges from the assembly on fixed reads that they may be assembled with. The motivation is that fragmentation may be caused by a relatively small number of reads that are discarded during error correction, and if they overlap the ends of contigs they may be brought into the original assembly to bridge gaps.

In real sequencing projects, read coverage varies greatly and is typically dependent on sequence features. For example, a 45 nt inverted palindrome was correctly read only 3 times in the entire set of 50 nt long reads with 500X coverage (in the human BAC dataset). In some applications (e.g., single cell sequencing), the variation in coverage is even more extreme. The variations in read coverage affect the k-mer coverage $Coverage_k(i)$ defined as the number of reads covering the position i in the genome and fully containing the k-mer starting at this position. We define a *k-mer gap* as a longest sequence of consecutive genomic positions with $Coverage_k(i) < m$, where m is the multiplicity threshold. The fluctuations in read coverage create k-mer gaps and each such gap fragments the repeat graph, as shown in Figure 6.9. It is possible to reduce the fragmentation of the repeat graph using a lower k-mer multiplicity threshold during error correction, or further by simply constructing the de Bruijn graph on all available reads, as error correction filters out reads that cannot be fixed. On the dataset of 50 nt long reads (50X coverage) lowering the multiplicity threshold from 5 to 2, and further to 1, decreases the number of gaps in the genome coverage from 27 to 15, and to 9 correspondingly. Lowering the threshold increases the size of the de Bruijn graph, due to inclusion of a number of erroneous reads, resulting in possible miss-assemblies. Therefore, we choose the k-mer multiplicity threshold conservatively as a lower bound. The threshold of 5, used in error correcting the dataset of 50 nt long reads, filters 8.1%

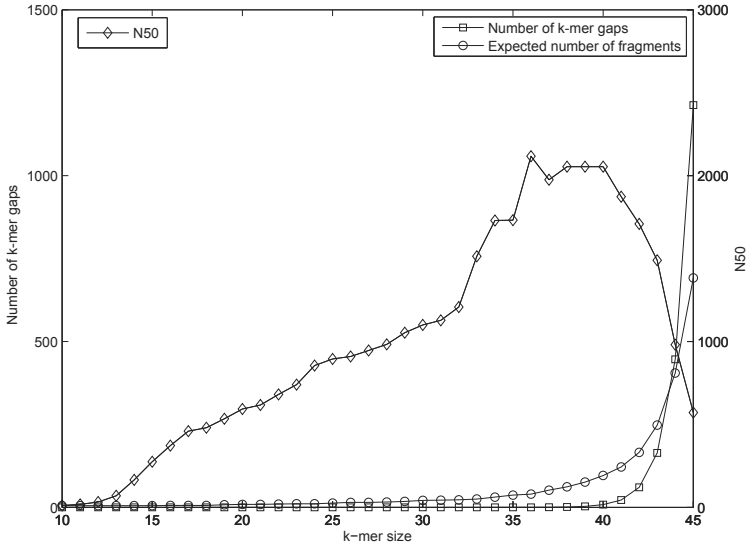


Figure 6.9 The choice of k-mer size affects the number of k-mer gaps (left Y-axis) in the the genome (human BAC). An increase in the k-mer size increases both the number of k-mer gaps and the number of connected components in OPTIMAL-ASSEMBLY(k) (negative effect). On the other hand, it increases the N50 size of OPTIMAL-ASSEMBLY(k) (positive effect). The N50 values are presented on the right Y-axis. For the Human BAC the optimal k-mer size is under 40 bases.

of original reads resulting in a high quality set of reads that covers the majority of the genome (99.6%). This allows us to drastically reduce memory constraints relative to no thereshold, and operate on a more simple graph; the graph on reads without a k-mer multiplicity threshold has 320,261 vertices and 338,836 edges versus 10,060 vertices and 11,464 edges in the de Bruijn graph on error corrected data. Table 6.6 shows the number of gaps in the genome coverage formed by reads on different stages of our algorithm.

We examined the tradeoff between the k-mer size and repeat graph fragmentation by constructing the de Bruijn graph on k-mers with k ranging from 10 to 45 nt using perfect reads. The results are shown in Figure 6.9. The number of gaps grows similarly to that expected by the Poisson distribution of fragments if coverage is uniform, however it grows slightly more quickly indicating sample bias at the resolution of less than the size of a read.

Read length	Coverage threshold	Original reads	SA corrected reads	Threaded and Second Chance reads
BAC35	5	27	69	41
BAC50	5	27	59	56
simBAC100	5	14	17	14

Table: 6.6 The number of k-mer gaps at different stages of the EULER-USR assembly ($k = 20$). The increase in the number of k-mer gaps after the error correction is mainly caused by discarding reads.

6.G EULER-SR runtime

The EULER-SR running time can be estimated as follows. The input to EULER-SR is given as a set of t reads of length n each of the overall length $N = t \cdot n$. Error correcting reads and the construction of the de Bruijn/repeat graph takes $O(N \cdot \log N)$ time [24]. Threading requires a depth first search starting at the edge on which the read begins. While the total number of edges searched per read may be high, it is usually small for typical repeat graphs with small vertex degrees (like most repeat graphs). Therefore, the overall running time is typically $O(N \cdot \log N)$.

The Velvet assembly ran considerably faster than EULER-SR: 7 minutes compared to 48 minutes. Since the data required roughly a week to generate, we do not anticipate assembly time to be a bottleneck in bacterial sequencing projects.

6.H Detailed description of how mate-paths are used to resolve repeats

Once mate-pairs have been transformed into mate-reads, we use the resulting *mate-paths* to resolve the repeats and simplify the repeat graph (compare with the Eulerian Superpath Problem discussed in [104]). Mate-paths that begin and end on edges with sequences unique to the genome may be used to resolve repeats. To detect if an edge is (likely) unique, first consider a repeat graph $G = (V, E)$ constructed on a genome. We will use paths in the repeat graph defined by reads (read-paths) and mate-pairs (mate-paths) to label edges as unique or not (read-paths and mate-paths correspond to superpaths in [104]). The motivation for determining which edges are unique

are from solutions to the *Chinese Postman Path* (CPP) discussed in [24]. A CPP P^{CP} is a path of minimal length that visits every edge at least once. Edges that are visited multiple times on the CPP are not unique. The path P^{CP} may be replaced by an Eulerian path if edges are replaced by multiedges with a count equal to the number of times they are traversed in P^{CP} [104]. Let $\mathcal{P} = (P_1, \ldots P_n)$ be a set of arbitrary subpaths from P^{CP}, but for the sake of simplicity, assume that no path in $\mathcal{P}$ is fully contained by any other path in $\mathcal{P}$. If an edge e is represented by a sequence that is unique in the genome, then only one path path in $\mathcal{P}$ may begin with e, and only one path from $\mathcal{P}$ may end with e. When more than one path from $\mathcal{P}$ begins with (ends with) e, there is more than one sequence in the genome that begins with (ends with) e, therefore e is not unique. The converse is not necessarily true; it is possible that only one path from $\mathcal{P}$ starts and ends at e because $\mathcal{P}$ is a set of arbitrary subpaths from P^{CP}. We denote an edge as unique if the above condition holds given a set of paths $\mathcal{P}$.

In assembly projects, the P^{CP} is not known and the goal is to detect it using mate-pair information. Mate-paths that start and end on unique edges must be part of P^{CP}. To detect unique edges using reads, the set of sub-paths from the repeat graph are defined by read-paths $\mathcal{P}^R$ and mate-paths $\mathcal{P}^M$. A path that begins and ends on unique edges is a *resolving path*. To find resolving paths, let $\mathcal{P} = \mathcal{P}^R \cup \mathcal{P}^M$. Again, for simplicity, assume no path in $\mathcal{P}$ is fully contained by any other path in $\mathcal{P}$ (if not, $\mathcal{P}$ is further processed to remove such paths). For each edge e, $start(e)$ is the the number of paths starting at e, and $end(e)$ is the number of paths ending at e. When a path P from $\mathcal{P}$ begins at an edge e_s with $start(e_s) = 1$ and ends on an edge e_e with $end(e_e) = 1$, P is a resolving path, and may be used to link e_s to e_e in the assembly.

The repeat graph is transformed using resolving paths to route paths out of repeat tangles (similar to the *equivalent transformations* in [104]). Given a resolving path P begining on an edge (s_{start}, d_{start}), and ending on edge (s_{end}, d_{end}). The edge (s_{start}, d_{start}) is replaced by an edge (s_{start}, d_{end}), and edges (s_{start}, d_{start}) and (s_{end}, d_{end}) are removed from the graph. The sequence of (s_{start}, d_{end}) is set to the sequence of the path (e_{start}, P, e_{end}), and all reads from mate-pairs that supported P are mapped to the new edge. Each edge contains a list of reads that map to it in the assembly. Reads that supported path (e_{start}, P, e_{end}) are removed from edges along P, and re-mapped to the

new edge (s_{start}, d_{end}). When no reads map to an edge, the edge is removed from the graph.

There are two difficulties when labeling edges as unique: an edge corresponding to a duplicated sequence is marked as unique (false positive), and an otherwise unique edge may be considered duplicated (false negative). The false positive labels result from missing paths due to a lack of coverage, i.e. no end of a mate-pair is sampled from a duplicated sequence. When coverage is high, we found that the rate of misclassifications is very low when considering edges of a minimal length 200 for *E. coli*. The second misclassification arises when paths are added erroneously to the graph due to reads that contain sequencing errors. Although error correction is performed both prior to assembly, and in graph correction operations, errors typically remain in repeat regions. We found that the set of paths $\mathcal{P}$ contains very few erroneous edges, although only using $\mathcal{P}$ results in more false positive edges. As a result, we resolve tangles first using $\mathcal{P}^R \cup \mathcal{P}^M$, then only $\mathcal{P}^M$.

6.I Paired-end assemblies using EULER-SR and Velvet

Figure 6.10 illustrates the results of both Velvet and EULER-SR by mapping contigs to the *E.coli* genome.

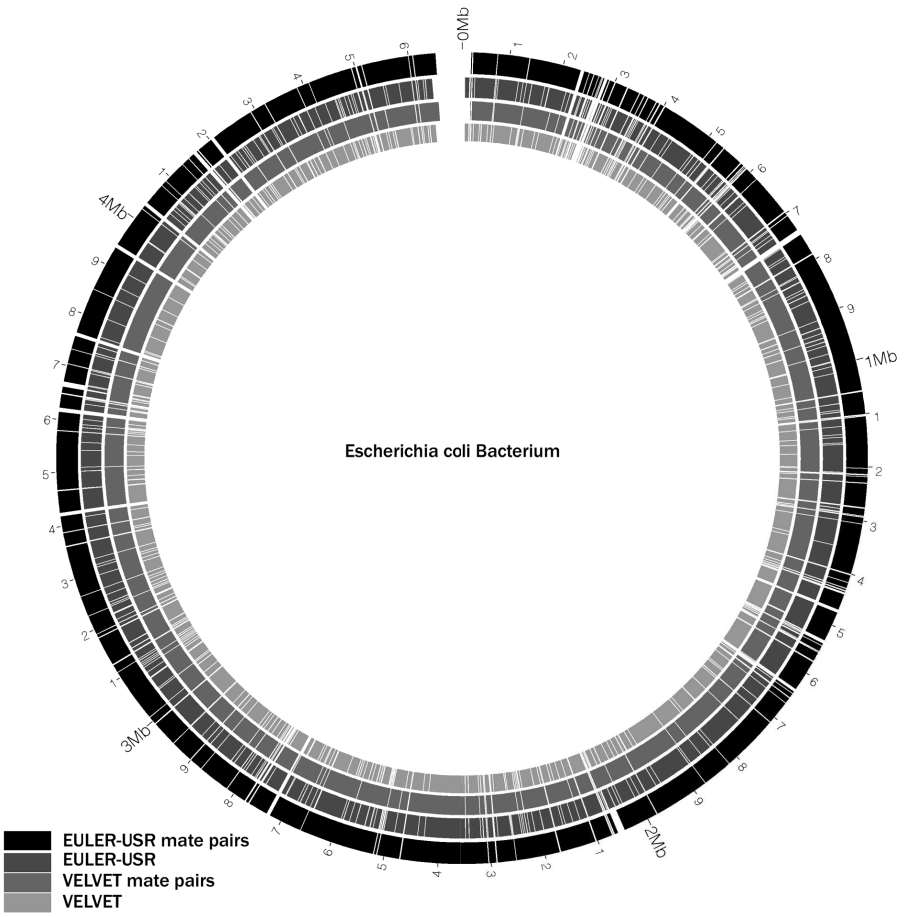


Figure 6.10 Comparison of contigs generated by EULER-SR and Velvet. The gap at the top shows where the reference genome was linearized.

7

Future Directions

The initial (and largely theoretical) studies regarding short read fragemnt assembly [23, 148] showed that contrary what some had believed, it was feasible to assemble genomes using short reads to similar quality of Sanger sequencing. When the inital short read data was released, several methods were quickly implemented and pubished [35, 146, 63], although they were written for perfect data or unrealistically high expectations of read quality. In the following year, four methods were published that are now considered "industrial" strength and were sophisticated enough to assemble large datasets of real data: Edena [53], Velvet [151], ALLPATHS [20], and EULER-SR [24]. The Edena, Velvet, and EULER-SR methods are in wide distribution and use within the short read sequencing community. The only method not shown thus far to assemble real reads is ALLPATHS. Our benchmarking tests have shown Velvet to produce the highest quality assemblies, second to EULER-SR. Although we have shown higher assembly quality using EULER-SR (for N50 values) than Velvet, others have reported the best quality by assembling data with all methods, and combining the result.[1].

There is a large community of users of EULER-SR: over 320 investigators have registered to download the source, and many have reported favorable results using it. In addition to the original application to assemblying bacterial genomes with EULER-SR, the versitility of high-throughput short read sequencing have lead to a large number of applications, many of which will benefit from *de novo* assembly.

[1] Shmuel Petrovsky, personal communication

7.A RNA-Seq Assembly

The field of functional genomics has been driven by microarryas for nearly the last decade. Microarrays, effectively massively parallel, miniaturized Northern blots on a chip, have several disadvantages. First, there is a large amount of noise with sequences with low expression levels due to cross hybridization, and volumes of publications have been dedicated to the statistical analysis of such data. More importantly, sequences not covered by probes are obviously left out from expression analysis.

The approach of RNA-Seq is to sequence cDNA samples in order to measure expression. Although the method is young, there are already protocols and publications showing that RNA-Seq analysis is comparative to microarrays [81, 86]. As sequencing is nearing the price-point where it is feasible to replace microarrays by sequencing, RNA-Seq will become more widespread.

RNA-Seq has the benefit that it is not limited to measuring expression of a probe set. Furthermore, since the sampling of cDNA is stochastic and fairly uniform, it is possible to quantify relative expression levels between two samples. The applications in [81, 86] measure RNA by mapping reads back to RefSeq. This analysis is limited because it does not identify splice forms, relative abundance of splice forms, and abberant splicing (such as in tumor genomes).

We have applied EULER-SR to the assembly of RNA-Seq from tumor cells in a collaboration with Catherine Quist and Arul Chinnaiyan. This analysis was able to assemble the noncoding sequence MALAT1, which was difficult to detect using arrays as well as mapping RNA-Seq reads because of its repetitive nature. Using Euler-SR the high level of the ncRNA was immediately apparent and Dr. Quist was able to use the assembled region to select an area to consider when calculating an expression level using RNA-seq.

Although there has already been success demonstrated with EULER-SR for *de novo* assembly of RNA-Seq data, many improvements may be made. First, EULER-SR performs error correction prior to assembly that assumes uniform of sampling of reads. The k-mer spectrum of reads is counted, and k-mers that appear above a certain threshold are considered to be correct. Since RNA-Seq samples uniformly from a wide

range of expression, the assumption of uniform coverag is not valid. Since the quality of the assembly depends largely on the preprocessing error correction, methods need to be developed to correct errors in reads sampled from a wide range of expression values.

Furthermore, although EULER-SR maintains a list of multiplicity of reads on every contig that is assembled, this information obscures information about abundance of splice variants. Every splice variant corresponds to a unique path in the repeat graph built on RNA-Seq reads. Robust methods remain to be developed to determine what splice variants are present in a sample, and their relative abundance using repeat graphs.

7.B Single Cell Sequencing

The current protocol to sequence a *species* is to collect DNA from a large sample of cells so that there is enough DNA to amplify before sequencing. For bacteria, the approach is to take a cell, and amplify it in a culture. For rapidly evolving species such as most prokaryotes, this may obscure how much genetic diversity is present. An alternative approach is Single Cell Sequencing [73] (SCS). In SCS, the genome is amplified using a random-primer based method called Multiple Displacement Amplification (MDA). While MDA is able to amplify enough DNA to sequence, it has the inherent drawback that some regions are amplified exponentially more than others.

Similar to the problem faced by *de novo* asembly of RNA-Seq data, *de novo* assembly of SCS must be able to account for widely varying coverage, rather than the uniform coverage one would expect when sequencing a large clonal sample.

7.C Large Genome Assembly

The initial release of "Next Generation" high-throughput sequencers produced enough reads for 50-100× coverage of bacterial genomes, reaching the minimal required coverage for assembling short reads. As of 2008 it is still cost-prohibitive for many labs to produce this coverage of larger, 100Mbase, genomes, and so the focus of *de novo* assembly has been on the assembly of bacterial genomes. Due to large investments from the National Human Genome Research Institute, sequencing throughput will likely increase again by several orders of magnitude (for "Next^2 Generation" sequencers), so

that it will be reasonable to sequence a mammalian genome in a single sequencing run. Such throughput will be useless if current methods are not able to assemble them.

EULER-SR has been shown to work on bacterial and yeast genomes. The next development is to use the current implementation to assemble the *Arabidopsis thaliana* and *Caenorhabditis elegans* genomes (both roughly 120Mbase).

7.D Metagenomic Assembly

EULER-SR has been used to assemble data from two separate oceanographic metagenomic samples: coral samples by Forest Rhower, and seawater samples by Vaughn Iverson in the Virginia Armbrust Lab at the University of Washington. The latter case was accompanied by proteomic analysis to map peptides to contigs assembled from ABI SOLiD reads. This analysis was performed on reads assembled without using mate-pair information. EULER-SR can currently use mate-paired reads to assemble bacterial genomes, but it remains to be shown how paired-ends will aid in assembling metagenomic samples.

References

[1] MD Adams. The genome sequence of *Drosophila melanogaster*. *Science*, 287:2185–2195, 2000.

[2] K Andries, P Verhasselt, J Guillemont, HW Gohlmann, JM Neefs, H Winkler, J Van Gestel, P Timmerman, M Zhu, E Lee, P Williams, D de Chaffoy, E Huitric, S Hoffner, E Cambau, C Truffot-Pernot, N Lounis, and V Jarlier. A diarylquinoline drug active on the atp synthase of mycobacterium tuberculosis. *Science*, 307:223–227, 2005.

[3] U Arnason, JA Adegoke, K Bodin, EW Born, YB Esa, A Gullberg, M Nilsson, RV Short, X Xu, and A Janke. Mammalian mitogenomic relationships and the root of the eutherian tree. *Proc. Nat. Acad. Sci.*, 99:8151–8156, 2002.

[4] V Bafna and PA Pevzner. Genome rearrangements and sorting by reversals. *SIAM J. Comput.*, 25:272–289, 1996.

[5] JA Bailey, Z Gu, RA Clark, K Reinert, RV Samonte, S Schwarz, MD Adams, EW Myers, PW Li, and EE Eichler. Recent segmental duplications in the human genome. *Science*, 297:1003–1007, 2002.

[6] N Bandeira, D Tsur, A Frank, and PA Pevzner. Abstract protein identification by spectral networks analysis. *PNAS*, 104:6140–6145, 2007.

[7] A Barski, S Cuddapah, K Cui, TY Roh, DE Schones, G Wei, Chepelev I, and K Zhao. High-resolution profiling of histone methylations in the human genome. *Cell*, 129:823–837, 2007.

[8] A Bashir, Y Chun, A Price, and V Bafna. Orthologous repeats and mammalian phylogenetic inference. *Genome Res.*, 15:998–1006, 2005.

[9] M Batzer et al. Alu insertion loci and platyrrhine primate phylogeny. *Molecular Phylogenetics and Evolution*, 35:117–126, 2004.

[10] S Batzoglou. The many faces of sequence alignment. *Brief. Bioinf.*, 6(1):6–22, 2005.

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

[12] M Blanchette, ED Green, W Miller, and D Haussler. Reconstructing large regions of an ancestral mammalian genome in silico. *Genome Res.*, 14:2412–2423, 2004.

[13] S Böcker. SNP and mutation discovery using base-specific cleavage and MALDI-TOF mass spectrometry. *Bioinformatics*, 19:44i–53i, 2003.

[14] S Böcker. Sequencing from compomers: Using mass spectrometry for DNA de novo sequencing of 200+ nt. *Journal of Computational Biology*, 11:1110–1134, 2004.

[15] G Bourque and PA Pevzner. Genome-scale evolution: Reconstructing gene orders in the ancestral species. *Genome Res.*, 12:26–36, 2002.

[16] G Bourque, EM Zdobnov, P Bork, PA Pevzner, and G Tesler. Comparative architectures of mammalian and chicken genomes reveal highly variable rates of genomic rearrangements across different lineages. *Genome Res.*, 15:98–110, 2004.

[17] I Braslavsky, B Hebert, E Kartalov, and SR Quake. Sequence information can be obtained from single DNA molecules. *PNAS*, 100:3960–3964, 2003.

[18] M Brudno, S Malde, A Poliakov, CB Do, O Couronne, I Dubchak, and S Batzoglou. Glocal alignment: finding rearrangements during alignment. *Bioinformatics*, 19:i54–62, 2003.

[19] P Buneman and F Hodson. *The recovery of trees from measures of dissimilarity.* Edinburgh University Press, 1971.

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

[21] G Cannarozzi, A Schneider, and G Gonnet. A phylogenomic study of human, dog, and mouse. *Plos Computational Biology*, 3:e2, 2007.

[22] A Caprara. Formulations and hardness of multiple sorting by reversals. *RECOMB 1999 Proceedings*, pages 84–93, 1999.

[23] M Chaisson, H Tang, and PA Pevzner. Fragment assembly with short reads. *Bioinformatics*, 20:2067–2074, 2004.

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

[25] MJ Chaisson, BJ Raphael, and PA Pevzner. Microinversions in mammalian evolution. *PNAS*, 103:19824–19829, 2006.

[26] YJ Chu and TH Liu. On the shortest arborescence of a directed graph. *Sci. Sinica*, 14:1396–1400, 1965.

[27] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2008.

[28] Mouse Genome Sequencing Consortium. Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420:520–562, 2002.

[29] The ENCODE Project Consortium. The ENCODE (ENCyclopedia Of DNA Elements) project. *Science*, 306:636–640, 2004.

[30] The ENCODE Project Consortium. Identification and analysis of functional elements in 1% of the human genome by the ENCODE pilot project. *Nature*, 447:799–816, 2007.

[31] TH Cormen, CE Leiserson, and RL Rivest. *Introduction to algorithms*. MIT Press, Cambridge, MA, 1995.

[32] R. de Rosa et al. Hox genes in brachiopods and priapulids and protostome evolution. *Nature*, 399:772–776, 1999.

[33] S Dear and R Staden. A sequence assembly and editing program for efficient management of large projects. *Nucleic Acids Research*, 19:3907–3911, 1991.

[34] TH Dobzhansky and AH Sturtevant. Inversions in the chromosomes of drosophila pseudoobscura. *Genetics*, 23:28–64, 1937.

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

[36] R Drmanac, I Labat, I Brukner, and R Crkvenjakov. Sequencing of megabase plus DNA by hybridization: theory of the method. *Genomics*, 4:114–128, 1989.

[37] J Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 61B:233–240, 1967.

[38] B Ewing, L Hillier, MC Wendl, and P Green. Basecalling of automated sequencer traces using phred. I. accuracy assessment. *Genome Research*, 8:175–185, 1998.

[39] H Fakhrai-Rad, N Pourmand, and Ronaghi. Pyrosequencing an accurate detection platform for single nucleotide polymorphisms. *Hum. Mutat*, 19:479485, 2002.

[40] J Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Massachusetts, 2003.

[41] L Feuk, AR Carson, and SW Scherer. Structural varaition in the human genome. *Nature Reviews Genetics*, 7:85–97, 2006.

[42] L Feuk, JR MacDonald, T Tang, AR Carson, M Li, G Rao, R Khaja, and SW Scherer. Discovery of human inversion polymorphisms by comparative analysis of human and chimpanzee dna sequence assemblies. *PLOS Gen.*, 1:e56, 2005.

[43] G Fischer, EP Rocha, F Brunet, M Vergassola, and B Dujon. Highly variable rates of genome rearrangements between hemiascomycetous yeast lineages. *PLoS Genet.*, 2:e32, 2006.

[44] H Fleischner. *Eulerian Graphs and Related Topics*. Elsevier Science, London, 1990.

[45] J Gallant and JA Maier, D nad Storer. On finding minimal length superstrings. *Journal Computer Systems Science*, 20:50–58, 1980.

[46] L Georgiadis. Arborescence optimization problems solvable by Edmonds' algorithm. *Theoretical Computer Science*, 301:427–437, 2003.

[47] SM Goldberg, J Johnson, D Busam, T Feldblyum, S Ferriera, R Friedman, A Halpern, H Khouri, SA Kravitz, FM Lauro, K Li, YH Rogers, R Strausberg, G Sutton, L Tallon, T Thomas, E Venter, M Frazier, and JC Venter. A Sanger/Pyrosequencing hybrid approach for the generation of high-quality draft assemblies of marine microbial genomes. *PNAS*, 103, 2006.

[48] ED Green. Strategies for the systematic sequencing of complex genomes. *Nature Reviews Genetics*, 2:573–583, 2001.

[49] N Gupta, S Tanner, N Jaitly, JN Adkins, M Lipton, R Edwards, M Romine, A Osterman, V Bafna, RD Smith, and Pevzner PA. Whole proteome analysis of post-translational modifications: applications of mass-spectrometry for proteogenomic annotation. *Genome Research*, 17:1362–1377, 2007.

[50] S Hannenhalli. Polynomial algorithm for computing translocation distance between genomes. In *Sixth AnnualSymposium on Combinatorial Pattern Matching*, volume 937, pages 162–176, 1995.

[51] TD Harris, PR Buzby, H Babcock, E Beer, J Bowers, I Braslavsky, M Causey, J Colonell, J Dimeo, JW Efcavitch, E Giladi, J Gill, J Healy, M Jarosz, D Lapen, K Moulton, SR Quake, K Steinmann, E Thayer, A Tyurina, R Ward, H Weiss, and Z Xie. Singlemolecule DNA sequencing of a viral genome. *Science*, 320:106–109, 2008.

[52] P Havlak, R Chen, KJ Durbin, A Egan, Y Ren, XZ Song, GM Weinstock, and Gibbs RA. The atlas genome assembly system. *Genome Research*, 14:721–732, 2004.

[53] D Hernandez, P Franois, L Farinelli, M Osteras, and J Schrenzel. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome Res*, 18:802–809, 2008.

[54] LW Hillier, GT Marth, AR Quinlan, D Dooling, G Fewell, et al. Whole-genome sequencing and variant discovery in *C. elegans*. *Nat Methods*, 5:183–188, 2008.

[55] DM Hillis. Sines of the perfect character. *Proc. Nat. Acad. Sci.*, 96:9979–9981, 1999.

[56] X Huang and A Madam. CAP3: a DNA sequence assembly program. *Genome Research*, 13:868–877, 1999.

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

[58] RR Hudson and NL Kaplan. Statistical properties of the number of recombination events in the history of a sample of DNA sequences. *Genetics*, 111:147–164, 1985.

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

[60] S Istrail, MS Waterman, EE Eichler, JC Venter, et al. Whole-genome shotgun assembly and comparison of human genome assemblies. *PNAS*, 101:1916–1921, 2004.

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

[62] JE Janeka, W Miller, TH Pringle, F Wiens, A Zitzmann, KM Helgen, MS Springer, and WJ Murphy. Molecular and genomic data identify the closest living relative of primates. *Science*, 318:792–794, 2007.

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

[64] Z Jiang, H Tang, M Ventura, MF Cardone, Marques-Bonet, X She, PA Pevzner, and EE Eichler. Ancestral reconstruction of segmental duplications reveals punctuated cores of human genome evolution. *Nature Genetics*, in press, 2007.

[65] J Jurka, VV Kapitonov, A Pavlicek, P Klonowski, O Kohany, and J Walichiewicz. Repbase update, a database of eukaryotic repetitive elements. *Cyto. Gen. Res.*, 110:462–467, 2005.

[66] D Karolchik, R Baertsch, M Diekhans, TS Furey, A Hinrichs, YT Lu, KM Roskin, M Schwartz, CW Sugnet, DJ Thomas, RJ Weber, D Haussler, and WJ Kent. The UCSC Genome Browser Database. *Nucleic Acids Res.*, 31(1):51–54, 2003.

[67] JD Kececioglu and EW Myers. Combinatorial methods for DNA fragment assembly. *Algorithmica*, 9:7–51, 1995.

[68] WJ Kent, R Baertsch, A Hinrichs, W Miller, and D Haussler. Evolution's cauldron: Duplication, deletion, and rearrangement in the mouse and human genomes. *Proc. Nat. Acad. Sci.*, 100(20):11484–11489, 2003.

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

[70] JB Kim, GJ Porreca, L Song, JM Gorham, GM Church, CE Seidman, and JG Seidman. Polony multiplex analysis of gene expression (PMAGE) in mouse hypertrophic cardiomyopathy. *Science*, 316:1481–1484, 2007.

[71] JO Kriegs, G Churakov, M Kiefmann, U Jordan, J Brosius, and J Schmitz. Retroposed elements as archives for the evolutionary history of placental mammals. *PLoS Biol.*, page e91, 2006.

[72] B Larget, JB Kadane, and DL Simon. A bayesian approach to the estimation of ancestral genome arrangements. *Molecular Phylogenetics and Evolution*, 36:214–223, 2005.

[73] R Lasken. Single-cell genomic sequencing using multiple displacement amplification. *Curr Opin Microbiol*, 10:510–516, 2007.

[74] M Li. Towards a DNA sequencing theory (learning a string). *Foundations of Computer Science, 1990. Proceedings*, 1:125–134, 1990.

[75] M Li, JH Badger, X Chen, S Kwong, P Kearney, and H Zhang. An information-baased sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17:149–154, 2000.

[76] RA Lippert, CM Mobarry, and BP Walenz. A space-efficient construction of the Burrows-Wheeler transform for genomic data. *Journal of Computational Biology*, 12:943–951, 2005.

[77] J Ma, L Zhang, BB Suh, BJ Raney, RC Burhans, WJ Kent, M Blanchette, D Haussler, and W Miller. Reconstructing contiguous regions of an ancestral genome. *Genome Research*, 16:1557–1565, 2006.

[78] O Madsen, M Scally, NC Douady, DJ Kao, RW DeBry, R Adkins, HM Amrine, MJ Stanhope, WW de Jong, and MS Springer. Parallel adaptive radiations in two major clades of placental mammals. *Nature*, 409:610–614, 2001.

[79] M Margiules, M Egholm, WE Altman, S Attiya, JS Bader, LA Bemben, J Berka, MS Braverman, YJ Chen, Z Chen, et al. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437:376–380, 2006.

[80] M Margulies and M Egholm. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437:326–327, 2005.

[81] J Marionil, C Mason, S Mane, M Stephens, and Y Gilad. Rna-seq: An assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, 18:1509–1517, 2008.

[82] P Medvedev, K Georgiou, G Myers, and M Brudno. Computability of models for sequence assembly. *Lecture Notes in Computer Science*, 4645/2007:289–301, 2007.

[83] ML Metzker. Emerging technologies in DNA sequencing. *Genome Research*, 15:1767–1776, 2005.

[84] R Mitra, J Shendure, J Olejnik, E Krzymansk-Olejnik, and G Church. Fluorescent in situ sequencing on polymerase colonies. *Anal. Biochem*, 320:5565, 2003.

[85] L Moret, BME Wang, T Warnow, and SK Wyman. New approaches for reconstructing phylogenies based on gene order. In *bioinformatics*, pages 165–173, 2001.

[86] A Mortazavil, BA Williams, K McCuel, L Schaeffer, and B Wold. Mapping and quantifying mammalian transcriptomes by rna-seq. *Nature Methods*, 5:621–628, 2008.

[87] S Murata, N Takasaki, M Saitoh, and N Okada. Determination of the phylogenetic relationships among pacific salmonids by using short interspersed elements (SINEs) as temporal landmarks of evolution. *Proc Nat Acad Sci*, 90:6995–6999, 1993.

[88] WJ Murphy, E Eizirik, SJ O'Brien, O Madsen, M Scally, CJ Douady, E Teeling, OA Ryder, MJ Stanhope, WW de Jong, and MS Springer. Resolution of the early placental mammal radiation using bayesian phylogenetics. *Science*, 294:2348–2351, 2001.

[89] WJ Murphy, DM Larkin, A Everts-van der Wind, G Bourque, G Tesler, L Auvil, JE Beever, BP Chowdhary, F Galibert, L Gatzke, et al. Dynamics of mammalian chromosome evolution inferred from multispecies comparative maps. *Science*, 309:613–617, 2005.

[90] WJ Murphy, PA Pevzner, and SJ O'Brien. Mammalian phylogenomics comes of age. *Trend. Gen*, 20:631–639, 2004.

[91] EW Myers. The fragment assembly string graph. *Bioinformatics*, 21(Suppl 2):ii79–ii85, 2005.

[92] EW Myers, GG Sutton, AL Delcher, IM Dew, DP Fasulo, MJ Flanigan, SA Kravitz, CM Mobanry, KH Reinert, and KA Renuington. A whole-genome assembly of Drosophila. *Science*, 287:2196–2204, 2000.

[93] P Ng, JJ Tan, HS Ooi, YL Lee, KP Chiu, MJ Fullwood, KG Srinivasan, C Perbost, L Du, WK Sung, CL Wei, and Y Ruan. Multiplex sequencing of paired-end ditags (ms-pet): a strategy for the ultra-high-throughput analysis of transcriptomes and genomes. *Nucleic Acids Research*, 34:e842006, 2006.

[94] M Nikaido, Rooney AP, and Okada N. Phylogenetic relationshps among cetartiodactyls based on insertions of short and long interspersed elements: Hippopotamuses are the closest extant relatives of whales. *Proc. Nat. Acad. Sci.*, 96:10261–10266, August 1999.

[95] M Nikaido, AP Rooney, and N Okada. Phylogenetic relationships among cetartiodactyls based on insertions of short and long interpersed elements: Hippopotamuses are the closest extant relatives of whales. *Proc Nat Acad Sci*, 96:9979–9981, 1999.

[96] N Okada. SINEs. *Curr Opin Genet Dev*, 1:498–504, 1991.

[97] JD Palmer and LA Herbon. Plant mitochondrial DNA evolves rapidly in structure,but slowly in sequence. *J. Mol. Evol*, 28:87–97, 1988.

[98] I Pe'er, N Arbili, and R Shamir. A computational method for resequencing long DNA targets by universal oligonucleotide arrays. *PNAS*, 99:15492–15496, 2002.

[99] H Peltola, H Söderlund, and E Ukkonen. SEQAID: a DNA sequence assembling program based on a mathematical model. *Nucleic Acids Research*, 12:307–321, 1984.

[100] PA Pevzner. l-tuple DNA sequencing: computer analysis. *J. Biomol. Struct. Dyn*, 7:63–73, 1989.

[101] PA Pevzner. *Computational Molecular Biology, an Algorithmic Approach*. MIT Press, Massachusetts, 2000.

[102] PA Pevzner. *Compuatational Molecular Biology*. MIT Press, Cambridge, MA, 2001.

[103] PA Pevzner, MY Borodovsky, and AA Mironov. Abstract linguistics of nucleotide sequences. ii: Stationary words in genetic texts and the zonal structure of DNA. *Journal of Biomolecular Structure and Dynamics*, 6:1027–1038, 1989.

[104] PA Pevzner and H Tang. Fragment assembly with double-barreled data. *Bioinformatics*, 17 Suppl. 1:S225–S223, 2001.

[105] PA Pevzner, H Tang, and G Tesler. De novo repeat classification and fragment assembly. *Genome Research*, 14:1786–1796, 2004.

[106] PA Pevzner, H Tang, and MS Waterman. An Eulerian path approach to dna fragment assembly. *PNAS*, 98:9748–9753, 2001.

[107] PA Pevzner, HA Tang, and GP Tesler. *De Novo* repeat family classification and fragment assembly. *RECOMB 2004 Proceedings*, pages 213–222, 2004.

[108] PA Pevzner and G Tesler. Genome rearrangements in mammalian evolution: lessons from human and mouse genomes. *Genome Research*, 13:37–45, 2003.

[109] PA Pevzner and G Tesler. Human and mousse genomic sequences reveal extensive reuse in mammalian evolution. *Proc. Nat. Acad. Sci.*, 13:7672–7677, 2003.

[110] PA Pevzner and G Tesler. Transforming men into mice: the nadeau-taylor chromosomal breakage model revisited. In *RECOMB 2003*, pages 247–256, New York, NY, USA, 2003.

[111] M Pop, S Salzberg, and M Shumway. Genome sequence assembly: algorithms and issues. *IEEE Comput*, 35:47–54, 2002.

[112] FP Preparata and E Upfal. Sequencing-by-hybridization at the information-theory bound: an optimal algorithm. *Jouanal of Computational Biology*, 7:621–630, 2000.

[113] A Price, NC Jones, and PA Pevzner. De novo identification of repeat families in large genomes. *Bioinformatics*, Supplement 1:i351–i358, 2005.

[114] BJ Raphael, D Zhi, H Tang, and PA Pevzner. A novel method for multiple alignment of sequences with repeated and shuffled domains. *Genome Research*, 14:2336–2346, 2004.

[115] A Reyes, C Gissi, F Catzeflis, E Nevo, G Pesole, and C Saccone. Congruent mammalian trees from mitochondrial and nuclear genes using bayesian methods. *Mol. Biol. Evol.*, 21:397–403, 2004.

[116] A Reyes, C Gissi, G Pesole, FM Catzeflis, and Saccone C. Where do rodents fit? evidence from the complete mitochondrial genome of sciurus vulgaris. *Mol. Biol. Evol.*, 17:979–983, 2000.

[117] MC Rivera and JA Lake. Evidence that eukaryotes and eocyte prokaryotes are immediate relatives. *Science*, 257:74–76, 1992.

[118] M Ronaghi. Pyrosequencing sheds light on DNA sequencing. *Genome Research*, 11:3–11, 2001.

[119] M Ronaghi, U Mathias, and P Nyren. Dna sequencing: a sequencing method based on real-time pyrophosphate. *Science*, 281:363–365, 1998.

[120] M Ronaghi, M Uhlén, and P Nyrén. A sequencing method based on real-time pyrophosphate. *Science*, 281:363–365, 1998.

[121] N Saitou and M Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol*, 4:406–425, 1987.

[122] F Sanger and AR Coulson. A rapid method for determining sequences in dna by primed synthesis with dna polymerase. *Journal of Molecular Biology*, 94:441–448, 1975.

[123] F Sanger, AR Coulson, GF Hong, DF Hill, and GB Petersen. Nucleotide sequence of bacteriophage lambda DNA. *Journal of Molecular Biology*, 162:729773, 1982.

[124] F Sanger, S Nicklen, and Coulson AR. Dna sequencing with chain-terminating inhibitors. *PNAS*, 74:5463–5467, 1977.

[125] D Sankoff, R Ceergren, and Y Abel. Genomic divergence through gene rearrangement. In *Molecular Evolution*, pages 428–438, 1990.

[126] D Sankoff, G Leduc, N Antoine, G Paquin, BF Lang, and R Cedergren. Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome. *PNAS*, 89:6575–6579, 1992.

[127] DE Schones and K Zhao. Genome-wide approaches to studying chromatin modifications. *Nat Rev Genet*, pages 9179–9191, 2008.

[128] M Schoniger and MS Waterman. A local algorithm for DNA sequence alignment with inversions. *Bull Math Biol*, 54:521–536, 1992.

[129] S Schwartz, L Elnitski, M Li, M Weirauch, C Riemer, A Smit, NISC Comparative Sequencing Program, ED Green, Hardison RC, and W Miller. MultiPipMaker and supporting tools: alignments and analysis of multiple genomic dna sequences. *Nucleic Acids Research*, 31:3518–3524, 2003.

[130] S Schwartz, WJ Kent, A Smit, Z Zhang, R Baertsch, RC Hardison, D Haussler, and W Miller. Human-Mouse Alignments with BLASTZ. *Genome Res.*, 13:103–107, 2003.

[131] 454 Life Sciences. 454 life sciences completes first whole genome sequence using first novel technology since 1977. Press release, 2003.

[132] J Shendure, GJ Porreca, NB Reppas, X Lin, JP McCutcheon, AM Rosenbaum, MD Wang, K Zhang, RD Mitra, and Church GM. Accurate multiplex polony sequencing of an evolved bacterial genome. *Science*, 309:1728–1732, 2005.

[133] J Shendure, NB Porreca GP, Reppas, X Lin, JP McCutcheon, AM Rosenbaum, MD Wang, K Zhang, RD Mitra, and GM Church. Accurate multiplex polony sequencing of an evolved bacterial genome. *Science*, 309:1728–1732, 2005.

[134] SS Singer, J Schmitz, C Schwiegk, and Zischler H. Molecular cladistic markers in new world monkey phylogeny (platyrrhini, primates). *Molecular Phylogenetics and Evolution*, 26:490=501, 2003.

[135] LM Smith, JZ Sanders, RJ Kaiser, P Hughes, C Dodd, CR Connell, C Heiner, SBH Kent, and LE Hood. Fluorescence detection in automated DNA sequence analysis. *Nature*, 321:674–679, 1986.

[136] R Staden. Sequence data handling by computer. *Nucleic Acids Research*, 4:4037–4052, 1977.

[137] M Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *J. Class.*, 9:91–116, 1992.

[138] A Sundquist, M Ronaghi, H Tang, P Pevzner, and S Batzoglou. Whole-genome sequencing and assembly with high throughput short read technologies. *PLoS One*, 2:e484, 2007.

[139] G Sutton, O White, M Adams, and A Kerlavage. TIGR assembler: a new tool for assembling large shotgun sequencing projects. *Genome Science Technology*, 1:9–19, 1995.

[140] MT Tammi, E Arner, E Kindlund, and B Andersson. Correcting errors in shotgun sequences. *Nucleic Acids Research*, 31:4663–4672, 2003.

[141] MT Tammi, E Arner, E Kindlund, and B Andersson. Correcting errors in shotgun sequences. *Nucleic Acids Research*, 31:4663–4672, 2003.

[142] J Tarhio and E Ukkonen. A greedy approximation algorithm for constructing shortest common superstrings. *Theoretical Computer Science*, 57:131–145, 1988.

[143] JW Thomas, JW Touchman, RW Blakesley, GG Bouffard, SM Beckstrom-Sternberg, EH Margulies, M Blanchette, AC Siepel, PJ Thomas, JC McDowell, et al. Comparative analyses of multi-species sequences from targeted genomic regions. *Nature*, 424:788–793, 2003.

[144] JC Venter et al. The sequence of the human genome. *Science*, 291:1304–1351, 2001.

[145] PJ Waddell and S Shelley. Evaluating placental inter-ordinal phylogenies with novel sequences including rag1, [gamma]-fibrinogen, nd6, and mt-trna, plus mcmc-driven nucleotide, amino acid, and codon models. *Mol. Phylo. Evol.*, 28:197–224, 2003.

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

[147] JL Weber and EW Myers. Human wholegenome shotgun sequencing. *Genome Research*, 7:401–409, 1997.

[148] N Whiteford, N Haslam, G Weber, A PrügelBennett, JW Essex, PL Roach, M Bradley, and C Neylon. An analysis of the feasibility of short read sequencing. *Nucleic Acids Research*, 33:171–176, 2005.

[149] Huelsenbeck JP Wong KM, Suchard MA. Alignment uncertainty and genomic analysis. *Science*, 319:473–476, 2008.

[150] KA Zaretskii. Construction of a tree on the basis set of distances between the hanging vertices. *Uspekhi Mat. Nauk*, 20:90–92, 1965.

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

[152] Z Zhang, S Schwartz, L Wagner, and W Miller. A greedy algorithm for aligning DNA sequences. *Journal of Computational Biology*, 7:203–214, 2000.

[153] D Zhi, B Raphael, A Price, H Tang, and PA Pevzner. Identifying repeat domains in large genomes. *Genome Biology*, 7:R7, 2006.

[154] E Zukerkandl and L Pauling. *Horizons in Biochemistry*. Academic Press, New York, 1963.