COMPUTATIONAL GENOMICS:

MAPPING, COMPARISON, AND ANNOTATION OF GENOMES

by

Serafim Batzoglou

B.S. Mathematics; B.S. Computer Science;
M.Eng. Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 1996

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy in Computer Science

at the
Massachusetts Institute of Technology
June 2000

Signature of Author.....
Department of Electrical Engineering and Computer Science
March 27, 2000

Certified by.....
Bonnie Berger
Samuel A. Goldblith Associate Professor of Applied Mathematics
Thesis Supervisor

Accepted by.....
Arthur C. Smith
Chairman, Committee on Graduate Students
Department of Electrical Engineering and Computer Science

ENG

Computational Genomics:
Mapping, Comparison, and Annotation of Genomes

by

Serafim Batzoglou

Submitted to the Department of Electrical Engineering and Computer Science
on March 27, 2000 in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

ABSTRACT

The field of genomics provides many challenges to computer scientists and mathematicians. The area of computational genomics has been expanding recently, and the timely application of computer science in this field is proving to be an essential component of the large international effort in genomics. In this thesis we address key issues in the different stages of genome research: planning of a genome sequencing project, obtaining and assembling sequence information, and ultimately study, cross-species comparison, and annotation of finished genomic sequence. We present applications of computational techniques to the above areas: (1) In relation to the early stages of a genome project, we address physical mapping, and we present results on the theoretical problem of finding minimum superstrings of hypergraphs, a combinatorial problem motivated by physical mapping. We also present a statistical and simulation study of "walking with clone-end sequences", an important method for sequencing a large genome. (2) Turning to the problem of obtaining the finished genomic sequence, we present *ARACHNE*, a prototype software system for assembling sequence data that are derived from sequencing a genome with the "shotgun" method. (3) Finally, we turn to the computational analysis of finished genomic sequence. We present *GLASS*, a software system for obtaining global pairwise alignments of orthologous finished sequences. We finally use *GLASS* to perform a comparative structure and sequence analysis of orthologous human and mouse genomic regions, and develop *ROSETTA*, the first cross-species comparison-based system for the prediction of protein coding regions in genomic sequences.

Thesis Supervisor: Bonnie Berger
Title: Samuel A. Goldblith Associate Professor of Applied Mathematics

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Number *Page*

ACKNOWLEDGEMENTS

My most special thanks to my advisor Bonnie Berger for her guidance, unwavering support, and collaboration on all the work described in this thesis. I gratefully acknowledge the guidance and collaboration of Sorin Istrail, Eric Lander, and Jill Mesirov. Sorin Istrail introduced me to physical mapping, and Chapter 1 is the result of our collaboration. I feel especially fortunate to have met and collaborated with Eric Lander, who deeply influenced my work, my knowledge of biology, and my devotion to genomics research. Eric Lander introduced me to the material of Chapters 2, 3, and 4. These chapters resulted from my collaboration with Bonnie Berger, Eric Lander, and Jill Mesirov. I especially thank my friend and colleague Lior Pachter, whose contributions directly enabled the completion of the comparative sequence analysis and gene recognition part of this work (Chapter 4).

I thank Bruce Birren, Ken Dewar, Daniel Kleitman, and Tomas Lozano-Perez, for helpful discussions. I thank Eric Banks, Wes Beebee, Valentin Spitkovsky, Ken Stanley, Tina Tyan, Brian Walenz, and Bill Wallis, for help and contributions to the research. I am grateful to Angelita Mireles, Amir Nashat, and Ken Stanley for their help in reading over and helping edit my thesis. I thank my family, and especially my sister Evi, for their support. Thank you all.

Για τον Συλβεστρο και τη Λαμπρινη

OVERVIEW

The discovery of the DNA double helix in 1953 by James Watson and Francis Crick (Watson and Crick, 1953a, 1953b) led the way to an understanding of biology in molecular terms. Subsequently, since the discovery of techniques to sequence DNA in the late 1970s (Maxam et al. 1977; Sanger et al. 1977) the characterization and study of the genetic material of organisms at the sequence level has become an increasingly important tool in biology. Today major genomic projects are being undertaken at an accelerating pace, with the aim of sequencing and ultimately studying the genetic material of humans, animals, plants, bacteria, and in general a vast variety of living organisms. The Human Genome Project, an international effort to produce the complete sequence of human DNA at very high accuracy, is scheduled to complete by 2003 (Science, vol. 284, p. 1439). The mouse genome is next in the pipeline (Science, vol. 287, p.1179). Organisms whose genome has already been sequenced include tens of unicellular organisms (see Hurowitz, 1999 for a list of 15 such organisms), yeast *S. cerevisiae* (Oliver et al. 1992; Dujon, 1996), *C. elegans* (The C. elegans sequencing consortium, 1998), and *Drosophila* (Nature, vol. 403, p. 817; Science, vol. 287, p. 1374). Many more organisms are either being sequenced, or will be sequenced in the near future. Finished sequences are being annotated with information about gene boundaries, regulatory elements, and other important biological units. Inter and intra-species comparative analyses of homologous regions are providing insight into the biology and evolution of organisms (O'Brien et al. 1999).

Since the first sequencing projects, computational tools have been essential in genomics. Shotgun sequencing for instance, the prevailing method for determining the sequence of a genomic region, involves obtaining a large number of short random pieces (a few hundred nucleotides long) of the region, sequencing those pieces with the existing sequencing technology, and then assembling them using computers into the complete sequence of the genomic region. As the pace of genomic research has been accelerating, the contribution of computer science is becoming both more essential and challenging.

In this thesis we will present some contributions of a computational/mathematical nature applied to the different stages of a genomic project. These stages include: planning a genome sequencing project, assembling the sequencing data into a complete genomic sequence, and finally comparing and annotating genomic sequences.

First, we will present theoretical work on the problem of finding the minimum superstring of a hypergraph. This is an algorithmic problem motivated by the biological problem of physical mapping, *i.e.* obtaining a map of the locations of clones for the purpose of then selectively sequencing them. We will study the computational complexity of this problem, and present some algorithms that provide constant approximations under certain conditions. We test our main algorithm on simulated random data.

Second, we will present a study of the "walking" approach to whole genome sequencing. Walking with clone-end sequences (Venter et al. 1996) is an important approach to genome sequencing, providing an alternative to either physical mapping-based clone-by-clone sequencing, or whole genome shotgun sequencing. We will present a mathematical model and computer simulations predicting the performance of a sequencing project based on walking with clone-end sequences. The purpose of our modeling work is to clarify important tradeoffs in planning to sequence a genome by this method. We also present a method for cutting dramatically the inefficiency of redundant sequencing, by using a second library of shorter clones.

Third, we will present some tools for automated assembly of shotgun sequencing data. We will present a hashing system for efficiently obtaining the adjacency matrix of similarity of a large collection of sequencing reads. Building on this, we will describe *ARACHNE*, a system for assembling shotgun sequencing reads into long layouts that we call supercontigs. We test a prototype implementation on the human chromosome 22, as well as on other sequences, with encouraging results. We briefly describe future improvements to *ARACHNE*, with the ultimate goal of having a system that can comfortably assemble shotgun data of a complete mammalian genome.

Finally, we will develop tools for comparison and annotation of large genomic regions of homologous DNA from two species. The impeding availability of vast amounts of unannotated human and mouse genomic data motivate this part of the work. The whole human genome will be available by 2003 while large regions of the mouse genome are being sequenced (Science, v. 284, p. 1906-1909, 1999; Science, v. 286, p. 210, 1999; Science, vol. 287, p. 1179, 2000) and in not

too long the complete mouse sequence will undoubtedly be obtained. The study of the human genome can benefit tremendously from comprehensive comparisons with the highly homologous mouse genome. Moreover, thinking ahead on future medical therapies and technologies based on genomics, they will likely be tested first on the mouse before being applied to the human. We will present *GLASS* (GLobal Alignment SyStem), a tool for aligning long orthologous genomic regions from related species. Using this, we will provide a comparative analysis of a number of orthologous genetic loci of human and mouse. Finally, we will present *ROSETTA*, a gene recognition program based on cross-species human and mouse genomic comparisons.

BACKGROUND

In the pages that follow we give a brief introduction to genomics, and outline the context of the contributions that we describe in the following chapters. We give basic biological background intended mainly for readers with expertise in computer science or related disciplines. Such readers may find it useful to also refer to more general texts such as (Lewin, 1996; Stryer, 1996; Lodish et al. 1998; Griffiths et al. 1993), and an introductory chapter on biology written for mathematicians (Lander and Waterman, 1995). For background more specific to the material in this thesis, we suggest the following selective reading: As general reading that is related to several parts of this thesis we would suggest Dujon, 1996; The C. elegans Sequencing Consortium, 1998; Lander, 1997; Lander and Waterman, 1988; Smit, 1995. For Chapter 1 on Physical Mapping, we would suggest Alizadeh et al. 1995; Booth and Lueker, 1976; Collins et al. 1995; Coulson et al. 1986; Greenberg and Istrail, 1995; Koop, 1995; Nelson and Speed, 1994. For Chapter 2, Venter et al. 1996; Lander and Waterman, 1988. For Chapter 3, Fleischmann, 1995; Green, 1997; Weber and Myers, 1997. For Chapter 4 we would suggest Altschul et al. 1990; Altschul et al. 1997; Batzoglou et al. 1998; Boguski et al. 1996; Burge, 1997; Burset and Guigo, 1996; Koop, 1995; O'Brien et al. 1999.

1. Genomes and the Genetic Code

DNA is a very long macromolecule composed of *deoxyribonucleotides*, which are small molecules containing a *base*, a *sugar*, and a *phosphate group*. The sugar contained in a deoxyribonucleotide is *deoxyribose*, indicating that it lacks an oxygen atom (*deoxy-*) that is present in a *ribose*. The base is a *purine* (*adenine* (A) or *guanine* (G)), or a *pyrimidine* (*thymine* (T) or *cytosine* (C)). Therefore DNA can also be thought of as a long string written in an alphabet of four letters, namely A, C, G, and T. DNA is actually a *double-stranded* molecule, where each nucleotide is bonded to its Watson/Crick complementary nucleotide (A to T, and C to G). Locally DNA assumes a helical shape, and therefore is called a *double helix*. James Watson and Francis Crick discovered the three-dimensional structure of DNA in 1953 (Watson and Crick, 1953a,b), and the mechanism of DNA replication was soon inferred. The complementary nucleotide chains

of DNA act as templates for each other during DNA replication. Therefore DNA is a large molecule containing information written in the alphabet {A, C, G, T}, capable of preserving and replicating this information. It is believed to contain (almost) all the information inherited by an organism from its parent(s).

The DNA of an organism is referred to as its *genome*. Genomes are stored inside cells. Organisms are broadly divided into *prokaryotic organisms* (or *prokaryotes*), and *eukaryotic organisms* (or *eukaryotes*). The major difference is that the genome of eukaryotes is sequestered in a *nucleus*, where it is protected by a *nuclear membrane* from the cytoplasm of a cell. The structure of the genome differs greatly between prokaryotes and eukaryotes, with eukaryotes usually having much longer genomes, exhibiting much higher repeat rates, and divided into a variable number of chromosomes. Multicellular organisms, *i.e.* organisms that have more than one cell, are usually eukaryotic. In multicellular eukaryotic organisms each cell has a nucleus with a copy of the DNA.

Figure 0.1. The Genetic Dogma

The precise sequence of bases in the genome carries the genetic information of the organism. *Genotype* is the genetic constitution of an organism. *Phenotype* is the expression of this genetic constitution, *i.e.* the physical characteristics of the organism, and depends on the genotype and on the environment. Different organisms have genomes that are very different in size and structure. Viruses in general have the shortest DNA genomes[1] ranging from 5kb (kilo-bases) up to 200kb. Bacteria genomes are longer, ranging from 500kb to 5Mb (mega-bases) and circular.

[1] In some viruses genetic material comes in the form of RNA instead of DNA.

The genome of yeast (a simple eukaryote) is 13.5Mb *long* (Sherman, 1997), contained in 16 pairs of chromosomes each ranging from 200kb to 2200kb in size.[1] The genome of the worm *Caenorhabditis elegans* (or *C. elegans*) is 97.3Mb long and is contained in 6 chromosomes (The C. elegans Sequencing Consortium, 1998). The genome of the fly *Drosophila melanogaster* is 165Mb long and is contained in 6 chromosomes (Rubin, 1996). The genome of a mammal such as human, or mouse, is believed to be around 3.5Gb (giga-bases) long. In some phyla, such as reptiles, birds, and mammals, genome sizes are similar within the phylum. But in the case of insects, amphibians, and plants, genomes vary widely in size within a phylum. Flowering plants have the greatest variation in genome sizes, ranging from smaller than 100Mb, to greater than 100Gb (Lewin, 1996).

Cells make *proteins*, which are the building blocks of a living organism. Proteins are synthesized according to the information encoded in *genes*, which lie in the genome. The expression of the information in a genome thus takes the following steps: genes are initially *transcribed* into *messenger RNA* (*mRNA*), which is an information-carrying intermediate in the protein synthesis mechanism. Messenger RNA in turn is *translated* into protein. A schematic is shown in Figure 0.0.1. RNA molecules are usually single-stranded, and are composed of adenine (A), cytosine (C), guanine (G), and uracil (U) instead of thymine (T). An mRNA *transcript* contains a single gene in eukaryotes, but in prokaryotes it usually contains multiple genes. Translation to protein takes place according to the *genetic code*, whereby three nucleotides are translated to one *amino acid*, which is a unit in a protein sequence, just like nucleotides are units in DNA and RNA sequences. There are 20 amino acids. Therefore the genetic code is redundant, as there exist 64 combinations of triplets of nucleotides. Three such combinations actually stand for the special terminator signals of translation, the *stop codons*: UAA (*ochre*), UAG (*amber*), and UGA (*opal*). One triplet (AUG) stands for the initiation of translation signal, as well as for the amino acid Methionine (Met). The full genetic code appears in Appendix A. Later in this section we give more details on the structure of the gene and on the processes of transcription and translation.

[1] We omit from this simple picture the fact that some of the DNA of an organism, and specifically yeast, is contained in mitochondria, plasmids, and even some double-stranded RNA viruses.

The human genome is believed to contain up to 100,000 genes. Probably by the end of the year this estimate will be more accurate, as human genome projects approach their conclusion. Other mammalian genomes are believed to contain a similar number of genes. Genes in other organisms vary. Table 0.1 shows estimated lengths of genomes and number of genes for a number of organisms (Meinke et al. 1998; O'Brien et al. 1999; Rubin, 1996; Lewin, 1996; The C. elegans Sequencing Consortium, 1998, Lin et al. 1999, Adams et al. 2000).

Organism	Length (Mbp)	Number of Genes
Human (and other mammals)	3,500	70,000-100,000
Drosophila (fruit fly)	120	13,600
Caenorhabditis Elegans (nematode)	97	19,000
Saccharomyces Cerevisiae (yeast)	13.5	5,800
Haemophilus Influenzae (bacterium)	1.83	1,738
Escherichia Coli (bacterium)	4.2	2,350
Arabidopsis Thaliana (plant)	120	27,000

Table 0.1. Estimated genome size and number of genes for some organisms.

2. Mapping and Sequencing a Genome

The complete DNA sequence of a genome is a powerful tool for studying an organism. Biological research in the 21st century will surely require obtaining the sequence of large numbers of important organisms, including many higher animals and plants with large genomes. Obtaining the genome sequences of living organisms is widely considered to be the first step towards a deeper understanding of genetics and biology in general. There is no simple biological experiment that can sequence an entire genome. There are several experiments though, that combined give much of the desired answer.

The first procedures that were able to determine the nucleotide sequence of small fragments of DNA emerged in the late 1970s (Maxam et al. 1977; Sanger et al. 1977). These procedures involved *gel electrophoresis*, an important biological experimental method. We refer the reader to standard introductory biology and biochemistry books (eg. Lodish et al. 1996). This is still the predominant method for sequencing DNA fragments. It can produce the sequence of a

500-1000 nucleotide *long* region off the end of a DNA fragment. The gel electrophoresis reaction is error prone, yielding around 1% incorrect nucleotides in the first 500 positions, and somewhat higher error rates in the subsequent positions. In addition, it is not possible to tell which of the two complementary strands of DNA is given by the reaction. The output of the experiment is called a *read*, and as we said is a 500-1000 long sequence on {A, C, G, T} of unknown orientation, and with ~1% errors.

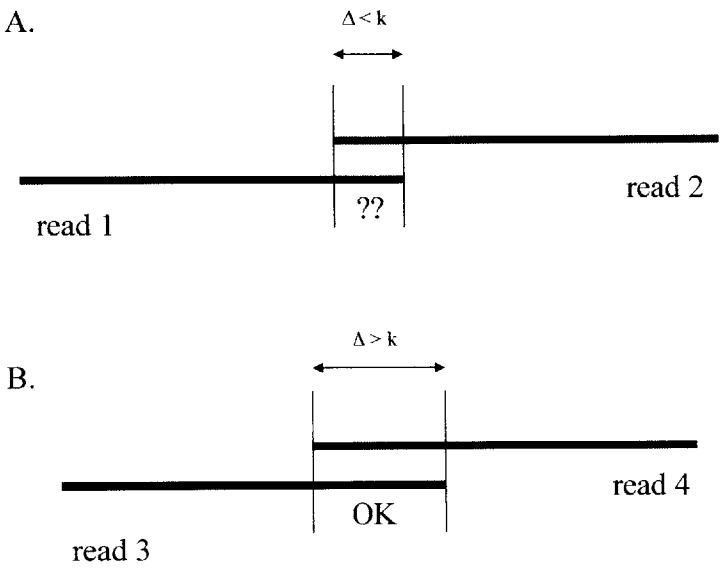

Figure 0.0.2. Ambiguous and unambiguous read overlaps. For the value of k, A. Reads 1 and 2 overlap by less than k, and therefore the overlap is ambiguous; B. Reads 3 and 4 overlap by more than k, and therefore are very likely to truly overlap in the genome.

The predominant method for characterizing longer regions is called *shotgun sequencing*, and was developed by Sanger's lab in 1982 (Sanger et al. 1982). The method involves (1) breaking several replicas of the DNA region of interest into many smaller pieces, at random; (2) obtaining reads using gel electrophoresis for a large number of those pieces; (3) finally detecting overlaps between the reads, and assembling them together using computers (*in silico*). In order for this approach to work, during step 2 a sufficient number of reads should be obtained. The

coverage with reads of a region of length L is defined to be C, where C×L is the sum of lengths of reads obtained in step 2. In such sequencing experiments, typical coverage values are 5-10, with coverage 10 being considered the "golden" quality standard.

Sanger's lab first applied shotgun sequencing to the genome of *bacteriophage λ* (Sanger et al. 1980, 1982). Subsequently genomes of large recombinant plasmids, large viruses, mitochondria, chloroplasts and bacteria were sequenced by this technique (Goebel et al. 1990; Oda et al. 1992; Ohyama et al. 1986; Fleischmann et al. 1995).

Fragment assembly is the computational problem of figuring out the source genomic sequence, given a collection of reads obtained by shotgun sequencing. In the absence of sequencing errors and repeats, fragment assembly is a trivial computational task, given a sufficiently large number of sufficiently long reads. Some back-of-the-envelope analysis demonstrates this claim. Let G be the genome length, N the number of reads, and assume for simplicity a constant read length l. Assume that the genome is a random string on {A, C, G, T}, with each letter occurring with probability 4^{-1} at each position independently of other positions. The probability that a given substring of length k (call it a *k-mer*) occurs in the genome is $\leq 4^{-k}\times G$. The probability it occurs twice is $\leq 4^{-2k}\times G^2$. There are 4^k k-mers, therefore the probability that any occurs twice in the genome is $\leq 4^{-k}\times G^2$. Letting $k = \lceil \log_4 G^2 + \log_4 10^3 \rceil$ the probability that there is a repeated k-mer is $\leq 10^{-3}$, which is very small. For $G = 3.5\times 10^9$, roughly the length of the human genome, it would suffice for k to be ≥ 37 in order to virtually ensure that two reads containing the same k-mer correspond to overlapping subsequences in the genome. This is shown in Figure 0.0.2. Then a simple algorithm would suffice to produce in linear time an assembly of the reads into large contigs: (1) create a hash table of all k-mers occurring in the reads, indexing the reads; (2) perform in any order, all possible pairwise "merges" of reads that contain a common k-mer. Step 1 above is linear in C×G, while step 2 is linear in C×N.

Factors that make fragment assembly considerably more challenging are (1) sequencing errors that result in reads with slightly different sequences in the region of overlap, and (2) the repetitive structure of genomes, especially for higher organisms and longer genomes. Currently *PHRAP* (PHragment Assembly Program, or Phil's Assembly Program after Phil Green, http://www.*PHRAP*.com) is the only generally available software system for the assembly of

shotgun data. It has been used successfully in most sequencing efforts in academia. It can handle routinely shotgun data of Bacterial Artificial Chromosome (BAC)[1] clones, which are usually 100-300Kbp long. It is believed to be capable of handling shotgun data of sequences that are at most a few million base pairs long, such as bacterial genomes (Lander, personal communication).

The major challenge when assembling shotgun data from large genomic regions is the presence of repeats in the genome. Different genomes exhibit different amounts and kinds of repetition. Bacterial genomes usually have very few repeats. Eukaryotic genomes usually exhibit a considerable amount of repetition. *Low complexity repeats* are regions of DNA that are extremely high in purines (A and T), or pyrimidines (C and G), or A and C, or G and T, or regions that contain *microsatellite* repeats,[2] or simply regions that are extremely rich in a particular nucleotide. Such regions clearly deviate considerably from a random sequence on {A, C, G, T}. In addition, there are common families of repeats. One example, *Alu*, are small repeats of length around 300 that tend to cluster, and are around 10% different between copies (Schmid, 1996; Batzer et al. 1996). *LINE*s (Long Interspersed Nucleotide Elements) are another family of repeats, of length between 500 and several thousand nucleotides (Smit and Riggs, 1995). The human genome contains roughly 1,000,000 Alu occurrences, and 200,000 LINE occurrences. Many other repeat families have been identified, including *MIR* and *LTR/Retroviral* (for a comprehensive review see Smit, 1995). Finally, possibly 25% of the genes in the human genome are repeated twice or three times within the genome, and there are also 43kb *long* tandem clusters of RNA pseudogene arrays, and 50-150kb *long* genome duplications (Myers, 1999). Such repetitions exhibit different degrees of fidelity between copies.

---

[1] Bacterial Artificial Chromosomes are a vector mechanism that can accommodate contiguous pieces of DNA that are fairly long (up to 300,000 *long* in general). Other vector mechanisms include plasmids, cosmids, Phage artificial chromosomes (PACs), and Yeast Artificial Chromosomes. Cloning of DNA is based on two key types of enzymes: *restriction enzymes* that are capable of cutting the DNA from any genome at occurrences of specific short subsequences of nucleotides, generating a well-defined set of fragments; and *DNA ligases*, capable of inserting the DNA fragments produced by restriction enzymes into replicating DNA molecules. Such replicating DNA molecules (vectors) include *plasmids* (often E-Coli plasmids) that can incorporate fairly short DNA sequences, on the order of 2,000-12,000 nucleotides long, *cosmids* that can incorporate longer sequences, on the order of ~40,000 long, *Bacterial Artificial Chromosomes (BACs)* that incorporate longer sequences of around 150kb long, and Yeast Artificial Chromosomes, that incorporate much longer sequences, on the order of 1Mb long. Different vector clone mechanisms exhibit various "hot" and "cold" spots in the genome, *i.e.* subregions of the genome that tend to be incorporated more frequently, or less frequently. The reason is restriction enzymes cut DNA in the occurrences of specific short sequences. This process is not entirely random, as DNA sequence does not obey much of the properties of a random string on four characters.

[2] Repeats of the form $(a_1 \ldots a_k)^n$ where $3 \leq k \leq 6$ and n is a very large number. The subsequence $a_1 \ldots a_k$ is repeated n times with a certain variation between repetitions.

In the presence of repeats, overlaps between pairs of reads do not unambiguously imply that the reads truly overlap in the genome. Figure 0.0.3 demonstrates how a repetition in the genome makes it hard to assemble the corresponding regions.

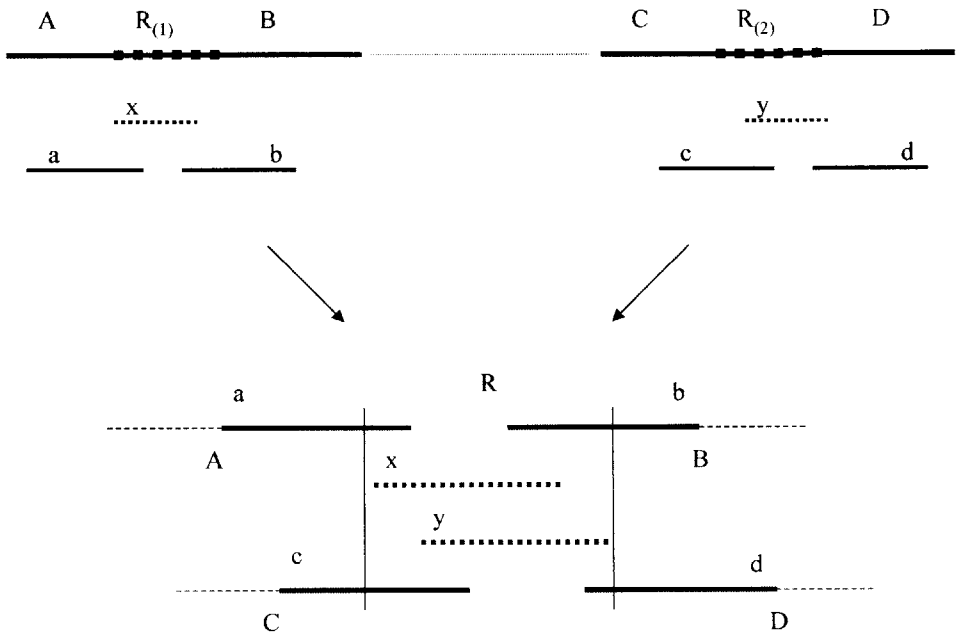

Figure 0.0.3. (Above) Repeat region R is flanked by unique regions A, B ($R_{(1)}$ occurrence) and C, D ($R_{(2)}$ occurrence). Reads x, y are completely inside the repeat occurrences, while reads a, b, c, and d are partially in the repeat, and partially in the unique regions. (Below) Correct assembly of the reads is a-x-b and c-y-d. However, wrong assemblies c-x-y-b, c-x-b, c-x-d, a-x-y-d, etc. are all consistent with the overlap picture.

Largely because of the presence of repeats, until recently it was widely believed that sequencing a large, repeat-rich genome with the shotgun approach and then accurately assembling the sequence is not feasible. For this reason, *clone-by-clone* approaches to sequencing a genome have been employed. According to these approaches, subsequences of the genome are inserted into vectors that can accommodate them, creating clones that can be reliably replicated. Enough clones are created to cover the whole genome with redundancy. Shotgun sequencing is

then applied to a subset of clones that has been selected so as to cover the entire genome with minimal redundancy. These approaches overcome the challenges of sequencing an entire repeat-rich genome using the shotgun method. Applying shotgun sequencing to each individual clone is much easier because (1) the subsequence is much shorter and consequently the computerized assembly of reads much smaller, and (2) a clone has a much simpler repetitive structure than the whole genome.

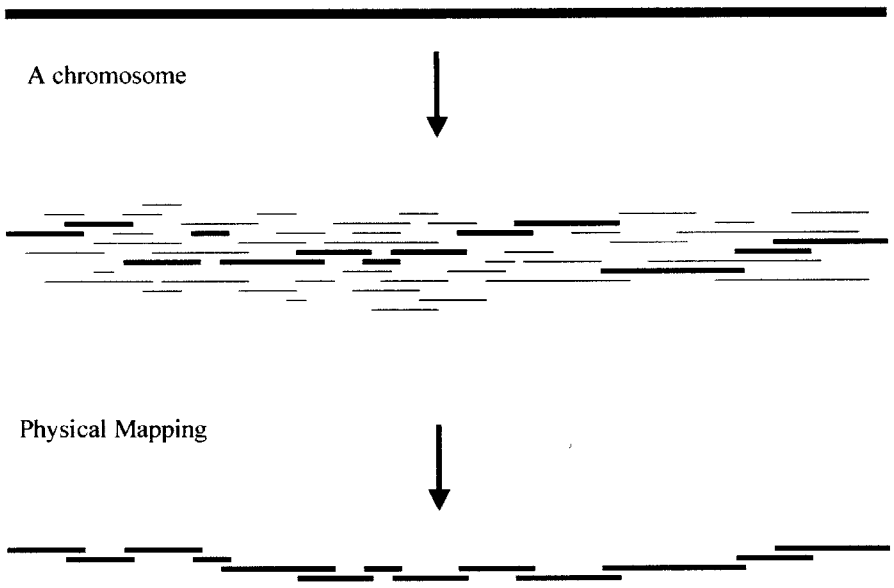

Figure 0.0.4. Clones, possibly from different kinds of vectors and of different sizes cover the chromosome. The clones are mapped so that their layout on the chromosome is known accurately enough to construct a tiling path. The clones in the tiling path are selected so as to minimize overlaps, while not allowing any gaps. These clones are eventually sequenced, providing the sequence of the chromosome.

Physical mapping is the predominant clone-by-clone approach to-date. (Other approaches have been proposed, see for instance Hudson et al. 1995; Venter et al. 1996; Batzoglou et al. 1999, and Chapter 2). According to this approach, as a first stage of a sequencing project, a large

library of clones is constructed covering the entire genome. The clones are mapped on the genome, *i.e.* their layout is known. Subsequently, a minimal *tiling path* of the clones is selected for the sequencing pipeline. Figure 0.0.4 shows an example of a mapped set of clones and a tiling path, for one contiguous piece of DNA (e.g. a chromosome). Note that in reality the DNA of a chromosome cannot be conveniently isolated. Therefore one cannot select and clone a random fragment off a specific chromosome. In order to sequence clones from a specific chromosome, the clones need to be mapped first to the chromosome.

The physical mapping techniques can be divided in two categories: *digestion* experiments, and *hybridization* experiments.

In digestion experiments a collection of clones is cut (*digested*) using one or more restriction enzymes, that each split the inserts at every occurrence of a characteristic substring of 4-8 nucleotides. Then the lengths of the resulting insert fragments are determined (usually using gel electrophoresis). Pairs of clones that exhibit similarities in the pattern of resulting fragment lengths are considered likely to overlap in the genome. Variations in the digestion technique include (1) single digests, in which one restriction enzyme is used; (2) double digests, in which two restriction enzymes are used, each cutting the inserts at occurrences of a different substring; (3) partial digests, where the target DNA is cut by one enzyme, but the experiment is performed multiple times, varying the duration of enzyme activity on each copy of the inserts. By varying the time the enzyme acts on DNA, larger or smaller sets of sites are recognized and cut by the enzyme.

In hybridization experiments, the presence or absence of a set of *oligonucleotide probes* is used to determine overlaps between insert clones. Oligonucleotide probes are small sequences of DNA that bind (*hybridize*) to their Watson/Crick complementary subsequences on a single-stranded insert clone. Probes detect overlap because when a set of probes hybridizes to the same pair of clones, the corresponding set of subsequences occurs in both clones. Hybridization experiments can be performed with simple oligonucleotide probes, or with STS probes. The STS

probes are pairs of 18 *long* substrings that are between 200 and 1,000 nucleotides apart in the insert. A region containing such a pair is called a *Sequence Tagged Site* (STS).[1]

Several kinds of experimental errors may occur with any of the above physical mapping techniques. In digestion experiments, typical errors include uncertainty in length measurements, multiplicity errors where the number of fragments of same lengths is miscounted, spurious or missing fragments, and other. Gel electrophoresis data on the lengths has typical error rates of 3% (Fasulo et al. 1997). In hybridization experiments, typical errors include *repeated probes*, *chimeric clones*, *false positives*, and *false negatives* (Greenberg and Istrail, 1995, Batzoglou and Istrail, 1999, Myers, 1999, *see also* Chapter 1). A false positive is the hybridization of a probe to a clone where it does not occur. A false negative is the absence of hybridization of a probe to a clone where it occurs. A chimeric clone is a piece of DNA coming from two different parts of the genome glued and appearing contiguous. A repeated probe is a probe that occurs at least twice on the mapped region, wrongly implying overlap between clones covering two different occurrences of the probe.

A fundamental question when constructing physical maps is to determine the number of clones needed to cover a genome. Lander and Waterman (1988) introduced a statistical model for answering this and related questions. Clones in the *Lander-Waterman model* are intervals of length L, each uniformly distributed along the genome which is a large interval [0,N] of length N. Clones cover the genome to depth (coverage) d = nL/N where n is the number of clones. The expected number of gaps in the genome (subregions not covered by any clone) is found to be $ne^{-d} + O(1)$, and the expected total gap length[2] is found to be Ne^{-d}.

Whole-genome shotgun sequencing is an alternative to the clone-by-clone approaches. An important variation of the technique, that considerably facilitates fragment assembly, is shotgun sequencing with *forward-reverse links* (Edwards and Caskey, 1991). According to this technique, reads are obtained from *both* ends of an insert (Figure 0.0.5). Moreover, inserts are size-selected so that the approximate distance of the pair of reads obtained from the ends of a fragment is

[1] STS probes are currently more popular because they are a more reliable and efficient means of obtaining the overlap information between clones (Myers, 1999).

[2] For the Human Genome, letting N = 3,500Mb, and given a library of n = 175,000 BAC clones of length around 200,000 each, so that the coverage d is 10, one would expect to get around 8 gaps, with total gap length 159Mb.

known. Plasmid inserts for instance, can be size selected to be approximately L *long*, with 2,000 < L < 12,000. One of the reads is always obtained by reading the forward strand of the fragment, and the other by reading the reverse complement strand. This is why the link between the reads is called a forward-reverse link. Because of imperfection in the experimental methods, forward-reverse links are inaccurate in two ways: (1) some links, typically 10% (Myers, 1999) are false with the respective reads coming from unrelated places in the genome; (2) distances between the reads are only approximate, with typical mean deviations in the 5-10% range. For simplicity we will also refer to forward-reverse links as *earmuff links*, or *earmuffs*.

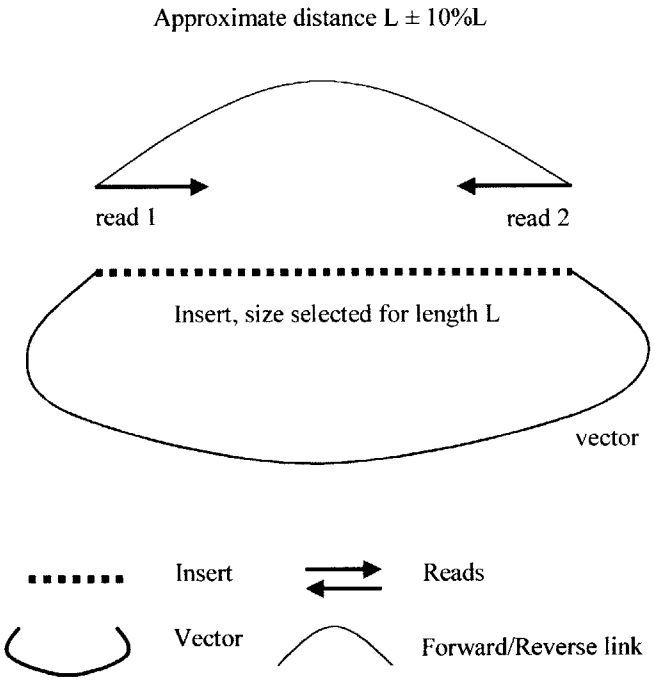

Figure 0.0.5. A forward/reverse link is obtained by reading off both ends of an insert.

Forward/reverse links can provide crucial information that aids in resolving repeats in the genome. Recall Figure 0.0.4, where regions A and B flank occurrence 1 of repeat R, and regions C, D flank occurrence 2 of R. Without any forward/reverse link information, it may not be

possible to deduce that A-R-B, C-R-D is the correct sequence, as opposed to A-R-D, C-R-B. Figure 0.0.6 shows how forward/reverse links can resolve the ambiguity.

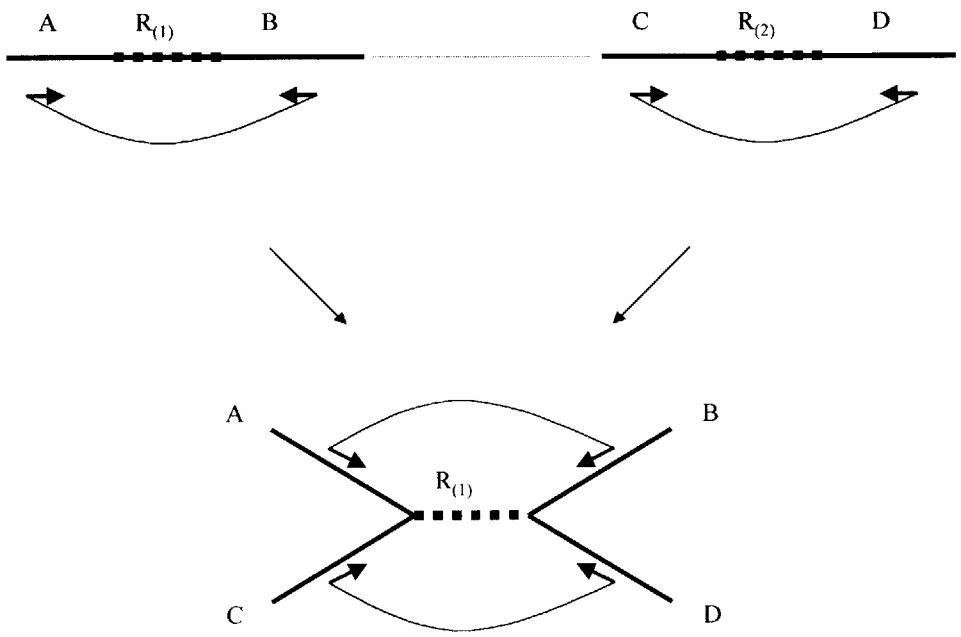

Figure 0.0.6. Forward reverse links between regions A,B, and C,D, resolve the ambiguity in the assembly. It is clear using the link information that regions A and B flank one copy of R, while regions C and D flank another copy of R.

Shortly after sequencing of an entire mammal genome became a realistic possibility, the US National Institute of Health, and the Department of Energy in collaboration with UK's Sanger center and other labs in Europe and Japan, announced the launch of the Human Genome Project (HGP) in 1990. A timetable was set for initially mapping, and subsequently sequencing the human genome with an objective to complete the project by 2005 (Collins and Galas, 1993). Subsequently, new goals were set for the HGP to complete by 2003, while also providing 90% of

the genome in a working draft by the end of 2001 (Collins et al. 1998). The HGP set a quality standard for finished sequence, of 99.99% — less than one error per 10,000 bases.

The Human Genome Project has been following the clone-by-clone approach. Weber and Myers (1996) proposed that the approach be shifted, to whole-genome shotgun sequencing. They provided a computer simulation demonstrating the feasibility of this approach. Their arguments in favor of shotgun sequencing included (1) better speed and cost; (2) detection of DNA polymorphisms;[1] (3) more complete coverage of the genome. Most notably they argued that pure shotgun sequencing could be performed in a centralized, efficient way, cutting the costs of the key procedures in a large factory setting. It would also avoid various steps needed in the clone-by-clone approaches, such as generation, storage, and tracking of large insert clones and smaller subclones. Finally it would avoid inefficient overlaps between clones that are being sent to the sequencing pipeline. On the other hand, Green (1996) advised against whole genome shotgun sequencing. The main argument against the shotgun method was the risk involved with performing all the costly sequencing work at once, and afterwards trying to assemble everything on a computer. Green (1996) argued that this approach may not yield a correct result, or a verifiable result, it may yield large-scale misassemblies of the sequence that are hard to detect and correct, and the finishing phase may prove more costly than the savings from previous phases.

In the past few years there has been a debate as to the strengths and weaknesses of each sequencing approach, mostly in view of the impending completion of the Human Genome Project. A privately owned company, *Celera*, in collaboration with academic partners, has been able to sequence the Drosophila genome by the shotgun approach combined with sequence obtained in the more traditional ways (Science, vol. 287, p. 767). Celera will partially sequence the human genome with shotgun, and combine the data with those of the Human Genome Project (Science, vol. 287, p. 1179). The few coming years will settle the debate regarding the efficiency and reliability of the various approaches for sequencing a large genome. Perhaps the optimal approach will prove to be a hybrid method of whole-genome shotgun sequencing using various vectors (yielding forward/reverse links of different lengths) and light shotgun sequencing of large inserts that cover the genome to some low depth (Lander, personal communication). The idea of

[1] Single-base differences between the DNA of two individuals of the same organism.

this approach would be to provide much of the needed sequencing through the whole-genome shotgun method, while using the light sequencing of random large clones as powerful additional linking information. The Drosophila genome was obtained with a hybrid method and Celera plans to obtain the Human Genome with a hybrid method. Finally, the mouse genome is in the pipeline to be sequenced by the academic sequencing community using a hybrid shotgun/clone-by-clone method (Science, vol. 287, p. 1179). The optimal combination of shotgun/hybrid for a large mammal genome is a very important open question (Lander, personal communication).

3. Genome Annotation

DNA is encoded into proteins according to the processes of transcription and translation (the Genetic Dogma). Not all of the DNA sequence is coding. In fact in mammals only a small percentage of the genome is coding. Segments of the genome that are translated comprise 75% of the yeast genome; they represent only about 3% of the human genome. One of the first steps in studying a genomic sequence is to identify the coding regions, and predict the encoded proteins. Given that transcription and translation are highly regulated processes, it is also useful to detect the precise locations of regulatory elements and other relevant subsequences that affect the processes of transcription and translation.

Before we discuss genomic annotation, we need to give a brief description of the processes of *transcription* and *splicing*. In the discussion that follows we will focus on eukaryotic organisms, and more specifically on mammals. In general bacteria do not have splicing, while the lower eukaryotes exhibit splicing to a lesser extent than higher ones.

RNA polymerase performs transcription by copying the DNA strand into an RNA strand that grows from 5' to 3'.[1] The copying is accomplished according to the rules of Watson-Crick base pairing. The source DNA is said to act as a *template* for copying into the newly formed RNA strand. The resulting RNA strand has the reverse complement sequence of the template. The RNA polymerase interacts with specific *promoter* sequences that are upstream of the

[1] DNA and RNA single stranded molecule ends are named 5' or 3' after the orientation of the 5' and 3' carbon atoms of the sugar ring. All known RNA polymerases synthesize chains directionally from 5' to 3'.

template and indicate the location where transcription should start. There are several kinds of promoter consensus sequences but in general there is no satisfactory computational method of recognizing promoters. Transcription is a most essential biological process, and its regulation is extremely important. Different organisms have various methods of regulating transcription. In addition to the promoter sequences, *enhancer* sequences can also regulate transcription. Enhancer sites are located at various distances from the transcription start; proteins bind at these sites and contribute to the regulation of RNA polymerase activity. Identification and annotation of these regulatory sequences by computational methods is a very important open problem. It would be a first step in understanding the mechanisms of regulation of specific genes.

Splicing occurs in RNA after transcription, and before translation. RNA before splicing is called *pre-mRNA*, while after splicing it is called *mRNA*. During splicing, certain enzymes (the *spliceosomes*) act upon pre-mRNA to delete certain segments called *introns* and connect together the remaining segments, called *exons*. The resulting sequence is used in translation, while the introns are thrown away.[1] Translation is executed by the *ribosome*, a large RNA/protein complex, and starts at an occurrence of the triplet AUG.[2] The translation start is usually surrounded by a weak consensus sequence, known as the Kozak consensus.[3] Translation proceeds directionally 5'→ 3'. Each triplet of nucleotides (called a *codon*) after the translation start is translated into an amino acid, according to the genetic code (Appendix A). This defines the *coding frame* of the gene. Translation stops at the first occurrence of a stop codon in the coding frame, which is as UAA, UAG, or UGA. Therefore, the translated part of the sequence contains no triplets UAA, UAG, or UGA in the coding frame. The leftmost untranslated part of the sequence, before the initiating AUG, is called the 5' untranslated region (5'-UTR), and the rightmost untranslated part, after the stop codon, is called the 3' untranslated region (3'-UTR). Abusing terminology, we will call the post-spliced segments *coding exons* or *noncoding exons*, depending on whether they are translated, or untranslated. Thus an exon may be fully coding, fully noncoding, or it may consist of a coding exon and a noncoding exon. A schematic is given in Figure 0.0.7. Three introns are

[1] Virtually all mRNA in vertebrate and insects are derived by splicing of pre-mRNA as described above. However, in some protozoans as well as in around 10-15% of the genes of the worm *Caenorhabditis Elegans*, a process of *trans-splicing* takes place whereby mRNA is produced by splicing together different RNA molecules (*see* Lodish, 1998).

[2] In humans the codons AUA and AUU also appear as initiation codons, while in mice AUC is also used. These occurrences are extremely rare and are usually ignored in computational recognition of genes.

[3] The Kozak consensus is the sequence CCRCCAATGG, where R stands for A or G.

spliced, generating 4 exons. In our terminology we have two 5' noncoding exons, three coding exons, and one 3' noncoding exon. The 5'-UTR for example, consists of the first exon and part of the second exon. In our terminology it consists of the first two noncoding exons.

Splicing of introns is controlled by the consensus sequence around the *splice sites*, which are the boundaries between introns and exons. The splice site at the 5' end of an intron is called the *donor* splice site, and the splice site at the 3' of an intron is called the *acceptor* splice site.

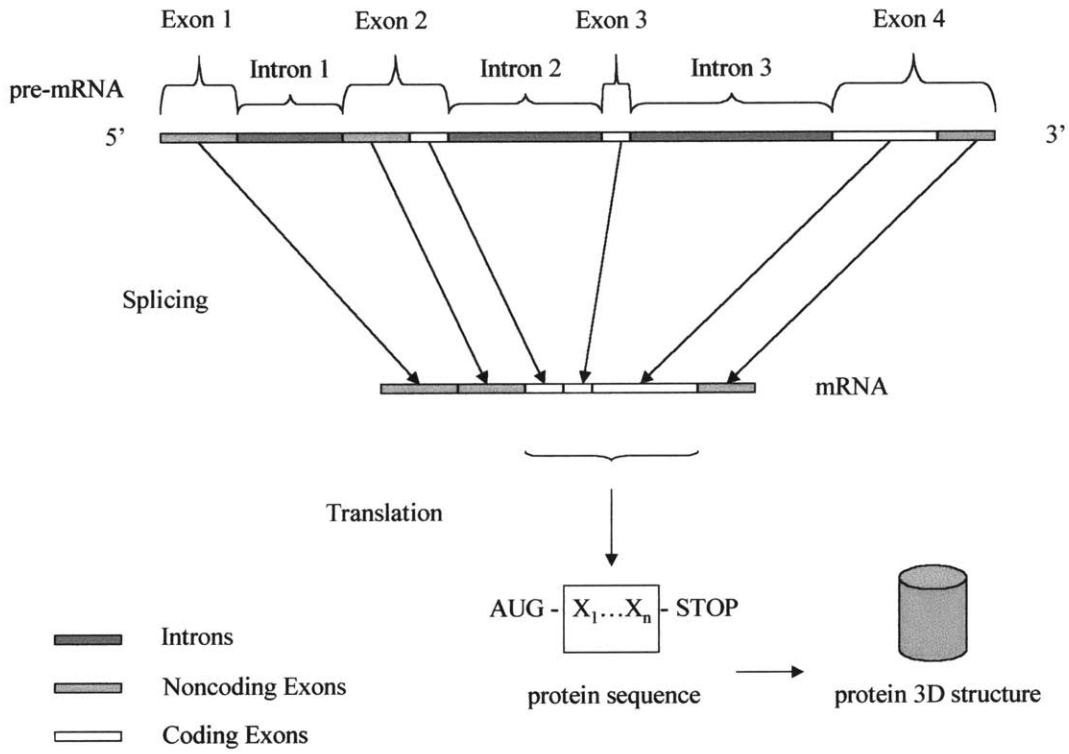

Figure 0.7. Splicing and translation. The pre-mRNA transcript undergoes splicing, where long segments (introns) are removed and the rest (exons) are glued together into the mRNA transcript. This in turn undergoes translation, where each triplet of codons after the first AUG is translated into an amino acid according to the genetic code, resulting in a protein sequence ($X_1 \ldots X_n$). This in turn is folded into a functional three-dimensional structure.

The donor splice site is generally characterized by a strong consensus of G**GU**RAGU where R stands for either A or G. The above consensus runs from the last position of the upstream exon, up to the 6th position of the intron. Less than half the splice sites obey the above consensus. However, the first two positions of an intron are GU in the vast majority of the introns.[1] Even though the few nucleotides adjacent to the GU start of an intron seem to be the most important in determining the splicing reaction, the region up to 20 base pairs into the intron seems to also play an important role. In certain introns, occurrences of GGG in that region play an important role in splicing (McCullough and Berget, 1997).

The acceptor splice site exhibits a weaker consensus of C**AG**R, where AG is always the 3' end of the intron.[2] Usually inside the intron and upstream of the acceptor splice site, lies a pyrimidine-rich region of length around 20, known as the *pyrimidine tract*. Further upstream often lies a *branch site*, which has the biological role of attaching to the 5'-end of the intron during splicing. The branch site lies usually immediately upstream of the pyrimidine tract. In yeast the branch site has a strong consensus of UACUA**A**C where the highlighted A is the point where the 5' end of the intron attaches, called the *branch point*. In human DNA there is a much weaker consensus of YNYURAY (Y stands for C or T, and N stands for any nucleotide).

Splicing is in general deterministic: for the majority of the genes, it is believed that the same genomic region always produces the same mRNA and protein products. However several genes can be spliced into a number of different variants depending on the environment or on the developmental stage of the organism. Those are called *alternatively spliced genes*, and the different splice sites that are used are called *alternative splice sites*.

The above discussion focused on the splicing of the vast majority of introns in pre-mRNA of insects and vertebrate. A different kind of splicing is exhibited in *group I* and *group II introns*. Group I introns are prevalent in nuclear *ribosomal RNA* (*rRNA*) genes of protozoans. Group II introns are found in organelles such as mitochondria and chloroplasts from plants and fungi. Group I and II introns exhibit self-splicing mechanisms where a spliceosome is not employed, and the RNA product catalyzes its own splicing reaction. The consensus sequences at the ends of

[1] GC is also observed in some cases.

[2] UAGR is also relatively common, while AAG is the most infrequent ending of an intron.

introns are vastly different for these groups of introns. We refer the reader to (Cech, 1990; Phizicky and Greer, 1993; Lodishet al. 1998) for further discussion on the splicing of group I, II introns, as well as to (Sharp and Burge, 1997) for splicing of some introns using a different spliceosome. A theory is that spliceosome-based splicing evolved from self-splicing reactions (Lodish et al. 1998).

Annotation of genomic sequence involves identification of the genes encoded in the sequence, including the boundaries of the regions that act as templates for transcription, the initiator and terminator of translation, splice sites, promoter and regulatory regions, and perhaps several other biologically important elements. Experimental methods exist for the accurate identification of most of the above elements, but it is generally understood that these methods are expensive, and too slow for the accelerating speed with which genomic sequence is being produced. For this reason computational methods play an essential part in genomic annotation. The identification of boundaries of coding exons, usually referred to as *exon prediction*, or *gene recognition*, is one of the most active areas of research in computational genomic annotation. Unfortunately computational methods are usually not as accurate as experimental methods, and computer-derived annotations often need to be experimentally verified.

Following annotation of sequence, a very important step in the study of a genome is functional annotation. Functional annotation is the process of assigning function to the genes that are expressed in the genome, discovering the mechanisms of regulation associated with different regulatory sites, and more generally assigning function to structural elements annotated in the genome. This step is very challenging and currently lags behind the sequencing and structural annotation steps (O'Brien et al. 1999).

4. Comparative Genomics

Genomes of different organisms exhibit similarities largely because of evolution from a common genome. For instance, the first mammal lived around 165 million years (*Myr*) ago. All modern mammals have a common ancestor that is dated between 65 Myr ago and 100-120 Myr ago. A common primate ancestor is believed to have lived around 60 Myr ago (O'Brien et al. 1999). Figure 0.8 shows the phylogeny tree for different well-known mammal orders (the more complete tree can be found in http://Ag.Arizona.Edu/tree/).

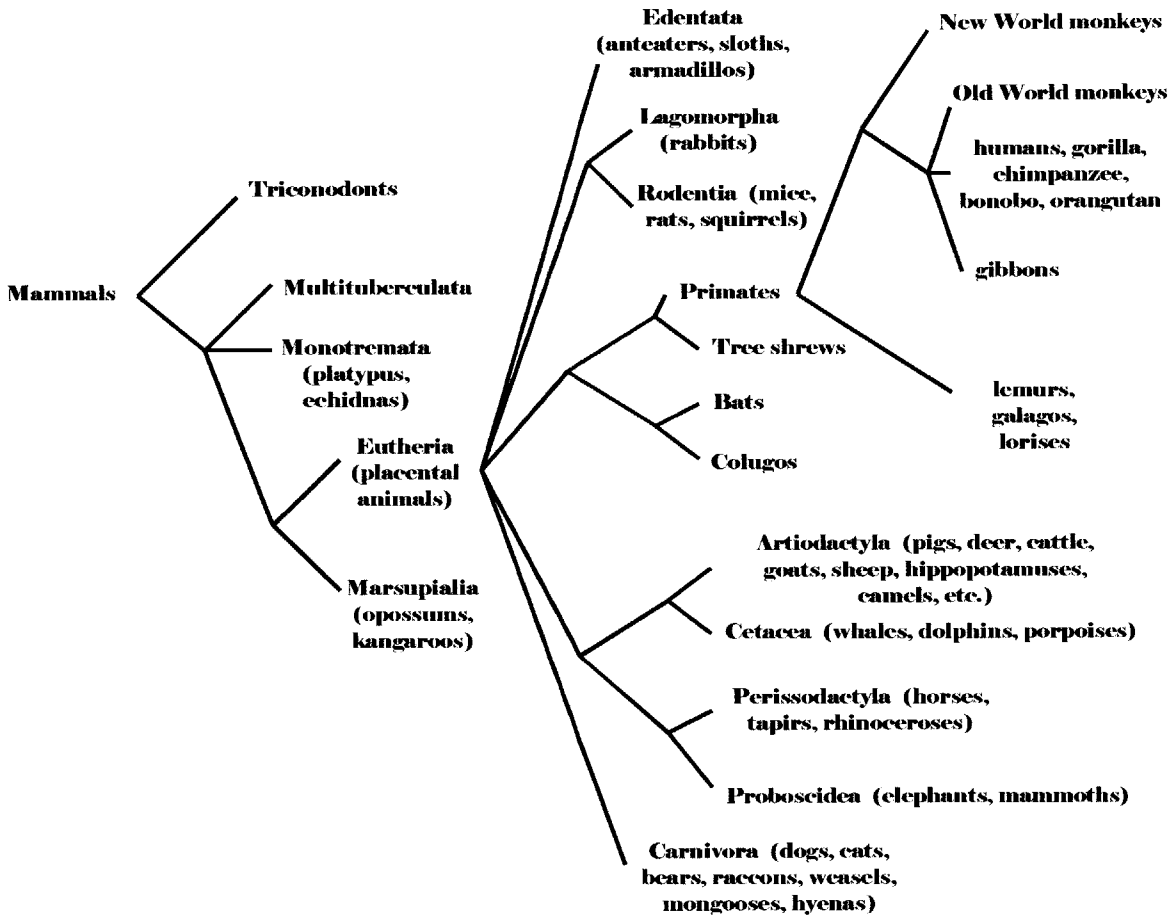

Figure 0.8. Part of the evolutionary tree for mammals. Refer to http://Ag.Arizona.Edu/tree/ for a fuller version of the tree of life.

Genomes change from generation to generation, by many different processes. The most local process is the random introduction of isolated mutations in the genome sequence that involve either base substitutions (change of a base pair among {AT, CG, GC, TA} to another), or insertions/deletions of a particular base pair (*indels*). Such mutations can be *silent* if they have no expressed effect (no effect in the phenotype) such as when a mutation occurs in an intron whose precise sequence presumably does not have any effect on the characteristics of the organism, or when a mutation occurs in the third position of a codon in an exon and does not change the encoded amino acid (refer to Appendix A). Other mutations change one amino acid but do not have any effect in the resulting protein activity (*neutral substitutions*). *Point mutations* are base

substitutions, and change only one amino acid in the protein. *Frameshift mutations* delete or insert a base pair, and that changes the frame of translation downstream from the mutation, resulting in a change in several amino acids. Frameshift mutations usually have a pronounced effect in the protein activity, often rendering the protein inactive or even deleterious.

More dramatic heritable genotypic changes include changes in the chromosome structure. These occur during meiosis where homologous regions in the genome have the opportunity to form Watson/Crick pairing, with subsequent large-scale exchanges of genomic regions. These changes broadly fall into four categories: *inversions*, *deletions*, *duplications*, and *translocations*. Inversions are 180-degree rotations of a usually large segment of a chromosome. They do not involve loss of genetic material, and therefore usually do not result in any phenotypic abnormality. They are found in around 2% of humans (Griffiths et al. 1993). Deletions are losses of part of a chromosome. In humans they usually cause syndromes of phenotypic abnormalities (eg. "cri du chat" syndrome[1]). Duplications can generate two copies of a region next to each other, or one in its normal location and the other in some other part of the same chromosome, or on a different chromosome. Duplications are important in genome evolution, because they supply additional genomic material that is capable of evolving new functions. Translocations are exchanges of parts between two different chromosomes. In the most common kind of translocation, reciprocal translocations, a segment from one chromosome is exchanged with a segment by another, resulting in two translocations (Griffiths et al. 1993). The most dramatic genotypic changes are changes in the number of chromosomes whereby chromosomes can be duplicated, lost, or even the whole genome can be duplicated. In mammals, chromosomal mutations that are kept by the species and propagated to later generations are extremely rare, so that only very few differences are apparent between the genomes of humans and different primates (O'Brien et al. 1988, Marshal Graves, 1998, O'Brien et al. 1999). The hypothetical primate ancestor's genome may have evolved into the human genome undergoing just seven translocations steps, while the cat's genome can be reorganized roughly into the human genome using only thirteen translocation steps. Comparative genomic analysis should eventually be able to resolve the full phylogeny tree among mammal orders (O'Brien et al. 1999).

[1] The "cri du chat" syndrome is a human genetic disease caused by a deletion at the end of the short arm of chromosome 5. The most characteristic phenotypic manifestation of the syndrome is the distinctive cries made by infants with this abnormality. Other manifestations include abnormally small head (microencephaly), a moonlike face, and mental retardation (Griffiths et al., 1993).

As we already mentioned single base mutations are generally silent when they occur in introns, or much of the intergenic regions. On the other hand, the effects of single based mutations can range from relatively minor to quite dramatic, if they occur within exons, region boundaries, and important structural elements such as promoters and regulatory elements. Most mutations do not have a beneficial effect on the phenotype, and the most dramatic ones are often deleterious. Such mutations do not tend to get propagated across generations because they are selected against during evolution. Thus, few of the mutations that occur in the most important regions of the genome get propagated to the next generations, while there is much less selective pressure against mutations occurring in less important regions such as introns or intergenic sequence. Mutations thus tend to accumulate over time in such regions at much higher rate than in the protein-coding and regulatory regions.

Because of selective pressure to preserve function, proteins and protein-coding DNA regions have remarkable sequence similarities across species. This is also true for regulatory regions such as promoters and for important signals such as consensus sequences in intron/exon boundaries. Within a particular species, genes frequently have inexact copies of themselves, often a few kilobases apart on the genome. Such gene similarities are believed to have arisen by duplications of an ancestral gene. Roughly half the genes in vertebrate are duplicated (Lodish et al. 1998), and form *gene families*. The resulting encoded proteins form *protein families*, often containing hundreds of members. The *immunoglobins* are a family of 500 proteins in several vertebrate species, while the *visual pigment proteins* are a family of 4 proteins in humans. *Pseudogenes* are copies of duplicated genes that have ceased to be expressed. Possibly silencing genetic events such as mutations on important transcription-control regions resulted in making such copies nonfunctional. The above facts point to the importance of sequence comparisons in numerous scientific problems. These include cross-species comparisons between close and distant relatives, and self-similarity comparisons within a single genome.

The mouse has been perhaps the most widely used model organism for understanding human biology, and for applications to medicine and drug discovery. Therefore there is already an abundance of data on the genetics, biology, and disease of the mouse. Moreover comparative analyses between mouse and human *orthologous* regions (regions that are hypothesized to have evolved from the same ancestral region) reveal very high sequence similarity (often above 80%)

in coding exons and regulatory regions (Lamerdin et al. 1994; Makalowski et al. 1996; Oeltjen et al. 1997; Makalowski and Boguski, 1998a,b, *see also* Chapter 4).

Hardison et al. (1997) argue in favor of sequencing the mouse genome for the purpose of detecting in the human genome the majority of the exons, and a significant number of regulatory elements that would not be detectable by computational methods in the absence of mouse homologous sequence. Both the academic sequencing community and private companies have shown interest in sequencing the mouse genome shortly upon completion of the human genome sequencing projects. The mouse genome is thus already in the sequencing pipeline of academic sequencing centers (Science, vol. 287, p. 1179, 2000), and a private sequencing company Celera (Science, v. 284, p. 1906-1909, 1999).

Similar efforts are taking place at a somewhat slower pace in plant genomics. Arabidopsis thaliana is the first plant whose genome will be sequenced by the target date 2003 (Gale and Devos, 1998). Sequencing of chromosomes 2 and 4 of Arabidopsis has recently been completed (Lin et al. 1999, Schuller et al. 1999). Comparative analysis of plant genomes will play an important role in the future for planning sequencing projects, and interpretation of genomic data for economically important grains such as wheat and maize, which both have genomes considerably longer than the human genome (Gale and Devos, 1998).

Chapter 1

PHYSICAL MAPPING WITH REPEATED PROBES

1. Introduction

The topic of this chapter is Physical Mapping, the most broadly used method for generating clones for the sequencing pipeline of a clone-by-clone sequencing project. We describe results on computational problems motivated by Physical Mapping. Our results are interesting mainly from a theoretical computer science perspective, but also have some potential applications to real Physical mapping experiments.

Physical mapping using hybridization data involves the construction of genomic maps based on the information contained in the clone-probe hybridization matrix. The mapping technique has to cope with computational difficulties that are specific to the hybridization data. There are errors that arise from the limitations in experimental accuracy. These include chimerism, false negatives and false negatives. Errors introduce specific combinatorial problems whose solutions could provide good mapping hypotheses. Usually these optimization problems are NP-hard and various heuristics – based on generalizations of the *Consecutive Ones Property* (C1P) (Booth and Lueker, 1976)– have been designed to cope with them e.g., (Alizadeh et al., 1995; Greenberg and Istrail, 1995). Another important combinatorial dimension of the mapping problem arises from the the fact that probes have multiple occurrences on the genomic region to be mapped. The literature dealing with algorithms for mapping in the presence of repeated probes is quite limited. In this chapter we study physical mapping with repeated probes, we identify some computational bottlenecks, and we propose algorithms that exhibit various degrees of measurable success.

The fundamental modeling paper of the area is the paper by Lander and Waterman (Lander and Waterman, 1988) in which the widely accepted Lander-Waterman model is introduced and analyzed; (*see also* Arratia et al., 1991; Green and Green, 1991; Nelson and Speed, 1994) for

further mathematical and statistical analyses. According to the Lander-Waterman model, clones are distributed uniformly along the genomic region, and probes are distributed according to a Poisson distribution.

The only published algorithmic work focussing on mapping with repeated probes is (Alizadeh et al., 1995), although further work devoted to the problem is in progress (Waterman, 1997; Shamir, 1997). In (Alizadeh et al., 1995) algorithmic strategies are proposed, based on the Lander-Waterman model by attempting to approximate the likelihood function, leading to NP-complete optimization problems that are reasonably tractable in practice. The algorithmic strategy proposed there uses local search 3-opt Lin-Kernigan type heuristics. Unfortunately approximation algorithms with a provable guarantee were not obtained. Based on this work, Karp (Karp, 1993) proposed the problem of designing approximation algorithms with guaranteed error bounds for the shortest superstring of a set collection. A superstring of a set collection is also called a *Hypergraph Superstring*. Each set in a set collection is a *hypernode*. The elements of each set are *hyperedges*. An element (*hyperedge*) then, connects two or more sets (*hypernodes*) if and only if the sets contain the element. The combinatorial problem of characterizing the superstrings of a hypergraph was introduced before (Ghosh, 1975; Lipski, 1976; Lipski, 1978) and it is notoriously difficult (Lipski, 1978; Erdos, 1993).

Kou proves in a paper devoted to information retrieval and file organization (Kou, 1977) that a variant of the C1P – modeling multiple storage of records – is NP-complete. In our terminology the result is that the Hypergraph Superstring Problem for strict Sperner hypergraphs is NP-complete. In (Lipski, 1978), non-tight upper and lower bounds were obtained for the hypergraph superstring length for the special case of the hypergraph being the power set of a finite set. Waterman (1995) gives a comprehensive overview of the problem.

A clone-probe hybridization matrix is a $0/1$ matrix with rows representing clones, columns representing probes, and a 1 in position (i, j) if and only if probe j is incident to clone i. Any permutation of the columns of such a matrix results in the same clone/probe incidence relationship. A collection of clones has the Consecutive Ones Property (C1P) (Booth and Lueker, 1976) if there is a permutation of the columns of the hybridization matrix that allows each row (clone) to be of the form $0 \cdots 01 \cdots 10 \cdots 0$ - in a consecutive ones form. The obvious biological rel-

evance of the C1P is that each clone spans a connected region of the genome. A clone-probe hybridization matrix containing "perfect" data, i.e., containing no errors and only unique probes, is a matrix that obeys the C1P. An important property for a heuristic mapping algorithm is to retrieve the C1P in the absence of errors (Greenberg and Istrail, 1995). This is one of the properties that our mapping algorithms achieve.

A meaningful simplifying assumption of the model is the *Sperner* property of a set collection: no set is included in the other. Indeed as the number of probes increases, the set of clones of the Lander-Waterman model has the Sperner property with high probability. The PQ-tree algorithm (Booth and Lueker, 1976) that retrieves the C1P uses a framework that hierarchically decomposes the initial collection of sets into subcollections that avoid sets included in unions of other sets.

The C1P property of a hybridization matrix ensures that there are no repeated probes. The *Sperner decomposition* of a set collection satisfying the C1P, and the optimal merging of sets in such a collection to obtain a PQ-tree are relatively easy computational tasks. Both tasks become computationally intractable for very sparse instances of data with repeated probes. Consider for example the intersection graph IG of a set collection. The vertices are the sets of the collection, and an edge exists between two vertices when the corresponding sets intersect. In the C1P case, the strict Sperner collections are sets of disjoint paths (SDP) in IG, while if probe repetition is allowed, these collections correspond to general graphs. These facts point out the importance of strict Sperner collections as building blocks in the hierarchical decomposition of the Hypergraph Superstring Problem. As we will see, both the Sperner decomposition as well as the optimal merging of the sets in a strict Sperner collection are MAXSNP- /NP-complete tasks.

In the above discussion the implicit assumption has been that a probe never appears more than once in a particular clone. This is a simplifying assumption that is justifiable probabilistically by the Lander-Waterman model, as the mean probe arrival time increases (probe repetition decreases). However, this property is not necessarily guaranteed. In fact the genome deviates from the Lander-Waterman model because of the repeat content of (particularly eukaryotic) genomes (Smit, 1995). An alternative model therefore, is to seek the minimal explanation of the hybridization data in the form of a *multiset superstring* that allows for possible repetition of probes in a

single clone. We prove that this problem is also MAXSNP-complete.

We present and test the *GREEDY-MERGE* algorithm that is based on Sperner decomposition of hypergraphs, with the following provable performance: (1) it retrieves the PQ-tree of all optimal zero-repetition superstrings; (2) on strict Sperner hypergraphs it is provably a 1.5625-approximation algorithm; (3) it provides a 2-approximation for hypergraphs with a restricted Sperner decomposition. The algorithm has cubic worst-case time complexity, and is much faster on sparse, data. When tested on simulated random data the algorithm approximated the length of the initial (correct) superstring within a factor of 1.1 in most problems involving 100-200 clones, 200-400 probes, and 1.5 to 4.9 average probe repetition. Please note that we did not compare the actual mapping of the clones predicted by the algorithm, with the original "true" map of the simulated data. We only report the ratio of the lengths of probe strings that the two maps provide. Moreover, the length of the initial superstring is not necessarily the minimum length. Therefore we cannot make any specific claims about how close our algorithm approximates the optimal superstring length in these experiments.

2. Background

2.1 Physical Mapping

DNA molecules are very long sequences over an alphabet of four letters, or nucleotides: $\{A, C, G, T\}$. Due to the large size of DNA molecules, the study of a genomic region involves breaking it into smaller pieces that can be sequenced by present technologies. *Physical Mapping* involves determining the true arrangement of the pieces on the initial genomic region, and then sequencing the smallest subset of pieces that cover the region. The *cloning* procedure incorporates the pieces of DNA into biological hosts. Each such copy is a *clone*. Through self-replication, a large number of copies of each clone are obtained. The result is a clone library containing many copies of pieces of the initial genomic region. The reconstruction process is based on data indicating "overlap" between clones. One method of detecting overlaps is through the hybridization of short sequences, called *probes*. Hybridization occurs when a probe sequence is complementary to a subsequence of a clone. If the probe has a unique occurrence on the initial genomic region and if

two clones are hybridized by the same probe then they overlap. This assumes ideal experimental conditions, i.e., no errors. So, unique probes detect overlap. However, in general probes are complementary to multiple places on the genomic region, so detecting overlap is ambiguous.

The information contained in the hybridization data can be summarized as follows. Let the clones be $\{C_1, \ldots, C_n\}$ and the probes be $\{P_1, \ldots, P_m\}$. Let the matrix H be defined by $H[i, j] = 1$ if probe P_j hybridizes to clone C_i, and $H[i, j] = 0$ otherwise. The problem we study is that of using hybridization data given in the matrix H to reassemble the clones such as to reconstruct the initial genomic region. Let us note that the process of breaking the DNA into pieces and selecting probes, even in a perfect cloning and hybridization experimental scenario, might result in loss of information. Therefore, we may not be able to obtain the exact reconstruction. To well-define the problem, we aim at obtaining the maximal mapping information consistent with H.

2.2 The Lander-Waterman Model

We will first define the Lander-Waterman model and then formulate a combinatorial problem in terms of hypergraphs, an appropriate framework for clone/probe hybridization data. Then superstrings are introduced in order to search for the minimum repetition of probes needed to explain the hybridization data.

The Lander-Waterman Model

1. A *clone* is an interval of length 1 contained in the interval $[0, N]$. The left endpoints of the clones are independent random variables, uniformly distributed over $[0, N - 1]$.
2. Probes $1, \ldots, m$ are distributed along the interval $[0, N]$ according to independent Poisson processes of rate λ. That is, a probe occurs at a short interval of length dx with probability λdx, and disjoint intervals are independent.

2.3 The Hypergraph Superstring Problem

Hypergraphs. A *hypergraph* is a pair $H = (X, \mathcal{S})$, where X is a finite set, and $\mathcal{S} = \{S_1, \ldots, S_m\}$ is a family of subsets of X. The sets S_i are called *hyperedges.* The following definitions apply to

hypergraphs as well to families of sets. A hypergraph is B-bounded if all of its hyperedges have at most B elements. A hypergraph is a *chain* if $\mathcal{S} = \{S_1, \ldots, S_m\}$ and $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_m$. A hypergraph is an *antichain*, or Sperner, if no S_i is included in S_j, for every $i, \neq j, 1 \leq i, j \leq m$. A hypergraph is *strict Sperner* if no hyperedge is included in the union of the other hyperedges, or equivalently if every hyperedge has a *characteristic* element.

A *Sperner decomposition* of a hypergraph $H = (X, \mathcal{S})$ is a decomposition of $\mathcal{S}$ into subfamilies of sets called *levels* or *layers* $\mathcal{S}_1, \ldots, \mathcal{S}_t$ such that: (1) the levels partition $\mathcal{S}$, i.e. $\mathcal{S} = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_m$ and $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$, $1 \leq i \neq j \leq t$; (2) $\mathcal{S}_i$ is a strict Sperner family of sets for every i, $1 \leq i \leq t$ and (3) $\bigcup \mathcal{S}_1 \subseteq \bigcup \mathcal{S}_2 \subseteq \cdots \subseteq \bigcup \mathcal{S}_t$.

Consider the clone-probe hybridization matrix of a Lander-Waterman process. Let P be the set of probes, and let $\mathcal{C} = \{C_1, \ldots C_m\}$ be the clones viewed as sets of probes. Then $H_{LW} = (P, \mathcal{C})$ is the associated hypergraph. According to the Lander-Waterman model, the arrivals of the left endpoints of the clones are distributed according to a Poisson process of rate $\frac{m}{N-1}$. If $|P|$ is large enough, with high probability no clone is a subclone of any other clone. Then H_{LW} is a Sperner hypergraph. The average number of probes per clone is $\lambda|P|$.

Multiset Superstrings. A string $\sigma = \sigma_1 \cdots \sigma_r$, is a multiset superstring of any subset of $U(\sigma) = \{S : 1 \leq \beta \leq \eta \leq r : S = \{\sigma_\beta, \sigma_{\beta+1}, \ldots, \sigma_\eta\}\}$.

Set Superstrings. A string σ is a set superstring (or simply, superstring) of any subset of $V(\sigma) = \{S : \forall \beta \leq i < j \leq \eta \quad \sigma_i \neq \sigma_j, S = \{\sigma_\beta, \ldots, \sigma_\eta\}\}$

For $S \in U(\sigma)$ or $S \in V(\sigma)$ we define $\beta_\sigma(S), \eta_\sigma(S)$ so that $S = \{\sigma_{\beta_\sigma(S)}, \ldots, \sigma_{\eta(S)}\}$. We say that σ *expresses* S if $S \in U(\sigma)$ ($S \in V(\sigma)$, also denoted by $S \in \sigma$. A multiset (set) superstring σ is *non-repeating* if no letter in σ occurs more than once.

Now we are ready to define our main computational problems:

The Hypergraph Set Superstring Problem: *Given* a Hypergraph $H = (X, \mathcal{S})$ *find* a superstring $\sigma = \sigma_1 \ldots \sigma_n$ for H of minimal length n.

The Hypergraph Multiset Superstring Problem: *Given* a Hypergraph $H = (X, \mathcal{S})$ *find* a multiset superstring $\sigma = \sigma_1 \ldots \sigma_n$ for H of minimal length n.

Remark. Let us remark that the corresponding Graph Superstring Problem, where the

hyperedges have exactly two elements can be solved in time linear in the number of edges in the graph. The minimum superstring coincides with the Eulerian path if the graph has such a path. In the general case, it coincides with the minimum size collection of Eulerian paths that cover all the edges. Our problem, the Hypergraph Superstring problem, is therefore a hypergraph generalization of the Eulerian path problem in graphs.

The Sperner Decomposition of a Hypergraph Problem: *Given* a Hypergraph $H = (X, \mathcal{S})$ and an integer $k > 0$, *decide* whether there exists a Sperner decomposition into k levels.

3. Computational Complexity Results

We show that the hypergraph set superstring, and the hypergraph multiset superstring problems are MAXSNP-hard. We prove these results with an L-reduction from *TSP(1,2)* on bounded degree undirected graphs. The same reduction proves both problems to be MAXSNP-hard. We are thus strengthening Kou's NP-completeness result (1977) by showing that the same problem is MAXSNP-hard, which implies that it is computationally intractable to approximate within better than a multiplicative constant of optimal. We also show that computing a minimal Sperner Decomposition of a hypergraph is a hard computational task: it is NP-complete to decide whether a two-level decomposition exists.

Theorem 1. The Hypergraph Set Superstring Problem and the Hypergraph Multiset Superstring Problem are MAXSNP-hard even for 5-bounded strict Sperner hypergraphs.

Proof. We use an L-reduction (intuitively a linear reduction, refer to (Papadimitriou, 1994)) from *TSP(1,2)* on undirected graphs, on instances where the graph formed by length-one edges has bounded degree. *TSP(1,2)* is the traveling salesman problem with distances $1, 2$. That is, given a complete graph G with edges of distance 1 and 2, find the shortest Hamiltonian path on the graph.[1] This problem has been shown to be *MAXSNP*-complete even if restricted to instances where the graph formed by the length-one edges has bounded degree (Papadimitriou and Yannakakis, 1993).

[1] That is, the shortest path that visits each node exactly once.

Let $H_G = (V, E)$ be a graph of bounded degree D specifying an instance of *TSP(1,2)*. That is, H_G contains the edges of cost 1 in the corresponding *TSP(1,2)* graph G. For every $v \in V = \{1, \ldots, n\}$, with associated edges $(v, u_1), \ldots, (v, u_d)$ where $d \leq D$, define hyperedge $S_v = \{v, \{v, u_1\}, \ldots, \{v, u_d\}\}$. The hypergraph H is then $(X, \mathcal{S})$ where $X = \bigcup_{v \in V} S_v$ and $\mathcal{S} = \{S_v | v \in V\}$. Clearly the above reduction can be performed in logarithmic space. Notice that the resulting set collection is Sperner because every set S_v has a distinguishing element $v \in S_v$. Moreover, $\forall v : |S_v| \leq D + 1$.

We will show that there is a Hamiltonian path on the graph G of *TSP(1,2)* of cost $n-1+k$ if and only if there is a (multiset, or set) superstring σ for $\mathcal{S}$ of length $m + k + 1$ where $m = |E|$. Since H_G is a graph of degree bounded by D, $m \leq D \times n$ is linear in n. This will establish that the above reduction is an L-reduction.

Say there is a Hamiltonian path of cost $n - 1 + k$. Since all edges have costs 1 or 2, we know the path uses $n - 1 - k$ edges from H and k edges of cost 2. Construct σ of cost $m + k + 1$ as follows: σ arranges the sets S_v in the order the nodes v are arranged on the path. Whenever an edge (u, v) in H_G is used on the path, S_u and S_v overlap in one element in σ. Then,

$$|\sigma| = \sum_{v=1}^{s} |S_v| - (n - 1 - k) = m + k + 1$$

Conversely, say that σ is a superstring of length $m + k + 1 = \sum_{v=1}^{n} |S_v| - (n - k - 1)$. Construct a path by reading in σ each vertex in the order it appears. Since σ is shorter than $\sum_{v=1}^{n} |S_v|$ by $(n - 1 - k)$ there is a total overlap of $(n - 1 - k)$ between the sets on the superstring. Since no two sets contain more than one common element, there are $(n - 1 - k)$ sets that overlap. These sets have a common edge. This establishes a total of $(n - 1 - k)$ edges from H_G used in the path, and hence a path of cost $(n - 1 + k)$. □

Theorem 2. Finding the minimum Sperner Decomposition of a Hypergraph is NP-complete. In particular, distinguishing between 2 and 3 levels for the minimum Sperner decomposition of a hypergraph is NP-complete, even for 3-bounded hypergraphs with size ≤ 1 hyperedge intersections.

Proof. Given a hypergraph $H = (X, \mathcal{S})$ and a partition of $\mathcal{S}$ into $\mathcal{S}_1, \mathcal{S}_2$, we can check efficiently the properties for a Sperner decomposition. Therefore, the Sperner Decomposition in k levels problem is in *NP*. We will show *NP*-hardness by a reduction from *3SAT*.

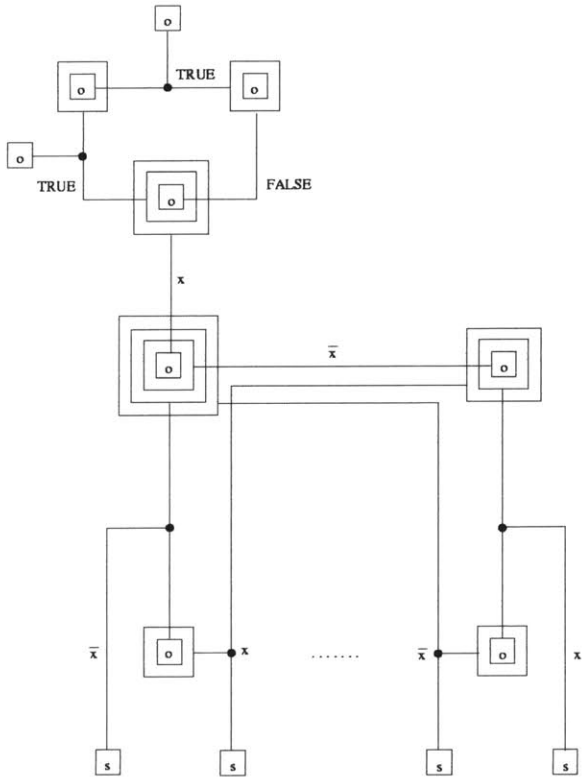

Figure 1: Gadget for truth assignment.

Let $\phi = \psi_1 \bigvee \ldots \bigvee \psi_m$ be a 3-CNF formula, with variables $x_1, \ldots, x_n$. We construct a hypergraph $\mathcal{S}_\phi$. Figure 1 shows the main part of the construction.

Two or three boxes connected by a line network correspond to one hyperedge. Any "*o*" contained in a box is a unique element in X. An "*o*" or "*s*" contained only in one box is contained only in one set. Such a set has to be in layer 1, because the union of layer 1 contains the union of layer 2. A set containing elements all belonging to sets in layer 1, has to be in a layer $\neq 1$.

Associate layer 1 with *TRUE* and layer 2 with *FALSE*. Then the top part of Figure 1 containing the three sets labeled *TRUE*, *TRUE*, and *FALSE*, should be self-explanatory. It follows that any two sets labeled x and $\bar{x}$ in Figure 1 are in different layers, in any 2-layer Sperner

decomposition.

Assign either all the x-sets, or all the $\bar{x}$-sets to layer 1 for each variable x, thereby constructing a truth assignment. Among the x-sets and the $\bar{x}$-sets, notice in Figure 1 that there are some containing an s-element. These sets are meant to correspond to literals in the clauses of ϕ.

For each variable x with k_x occurrences of literal x and k'_x occurrences of literal $\bar{x}$ construct k_x x-sets with an s-element, and k'_x $\bar{x}$-sets with an s-element. Finally, three s-elements collapse to one if and only if the corresponding literals are in the same clause ψ_i. Therefore there is one s-element for each clause.

Clearly a truth assignment satisfying every clause translates to a 2-level Sperner decomposition. Conversely, a 2-level Sperner decomposition correctly assigns truth value: $\forall x$ all the x-sets are in the same level, complement to the one with the $\bar{x}$-sets. Moreover, every s-element belongs to three sets at least one of which in level 1, thereby satisfying the corresponding clause. □

4. Approximation Algorithms

We describe a family of algorithms that repeatedly merge sequences of sets with the largest overlap, to produce a final solution that represents a collection of valid superstrings. The representation used is a PQ-tree that produces a family of superstrings of the same length. From this representation, any member can be extracted efficiently. We assume that the reader is familiar with the PQ-tree data structure. Below we outline the definition of a PQ-tree, as a representation for a collection of superstrings:

PQ-trees that produce superstrings: A PQ-tree is a tree made of P-nodes and Q-nodes. A P-node is either a finite set of symbols, or a parent node with a finite number of child PQ-nodes. A Q-node is a parent node with an *ordered* finite list of child PQ-nodes. A string σ is *produced* by a PQ-tree with top node P, a P-node, if (i) P is a finite set of symbols and σ lists those symbols once each in any order, or (ii) P is parent to PQ-trees $\{t_1, \ldots, t_k\}$ and $\sigma = \sigma_{\pi(1)} \cdots \sigma_{\pi(k)}$ where σ_i is produced by t_i, and π is a permutation on $\{1, \ldots, k\}$. A string σ

is *produced* by a PQ-tree with top node Q, a Q-node, if Q is parent to an ordered list of PQ-trees $[t_1, \ldots, t_k]$ and $\sigma = \sigma_1 \cdots \sigma_k$ or $\sigma = \sigma_k \cdots \sigma_1$ where σ_i is produced by t_i. We call a PQ-tree *proper* if all the strings it produces have the same length. For a proper PQ-tree rooted at node R, we define the *length* of R, $\mathsf{L}(R)$ to be the length of any produced string. We call a proper PQ-tree *non-repeating* if all the superstrings it produces are non-repeating. We call a PQ-tree a *superstring collection* for a hypergraph $H = (X, \mathcal{S} = \{S_1, \ldots, S_s\})$ if all produced strings are superstrings of H.

Lemma 1. Let $\mathcal{S} = \{S_1, \ldots, S_s\}$ be a Sperner system. Then

$$\forall S_i, S_j \in \mathcal{S} \quad (i \neq j) \implies ((\beta_{\sigma_{OPT}}(S_i) > \beta_{\sigma_{OPT}}(S_j)) \bigwedge (\eta_{\sigma_{OPT}}(S_i) > \eta_{\sigma_{OPT}}(S_j))) \bigvee$$
$$((\beta_{\sigma_{OPT}}(S_i) < \beta_{\sigma_{OPT}}(S_j)) \bigwedge (\eta_{\sigma_{OPT}}(S_i) < \eta_{\sigma_{OPT}}(S_j)))$$

Proof. Notice that if $\beta_\sigma(S_i) \leq \beta_\sigma(S_j)$ and $\eta_\sigma(S_i) \geq \eta_\sigma(S_j)$ then $S_j \subseteq S_i$ which is never the case in Sperner systems. The assertion of the lemma follows. □

The above lemma enables us to define an order $\prec_\sigma$ on the elements of a Sperner system $\mathcal{S}$:

Definition. Given a string σ, and any two sets C, D expressed in σ, we say that $C \prec_\sigma D$ iff $\beta_\sigma(C) < \beta_\sigma(D)$. For sets C, D of a hypergraph $H = X, \mathcal{S}$, we say that C is before D in σ, denoted by $C \lhd_\sigma D$, whenever $C \prec_\sigma D$ and $\nexists E \in \mathcal{S}(C \prec_\sigma E \prec_\sigma D)$. We say that C is next to D in σ, denoted by $C \bowtie_\sigma D$, whenever $C \lhd_\sigma D \bigvee D \lhd_\sigma C$.

Say that $S_1 \lhd_\sigma \cdots \lhd_\sigma S_s$. Denote by $\sigma_{[i,j]}$, $1 \leq i \leq j \leq s$ the substring $\sigma_{\beta_\sigma(S_i)} \cdots \sigma_{\eta_\sigma(S_j)}$, which is clearly a superstring for $\{S_i, \ldots, S_j\}$.

For sets C, D expressed in σ we say that $C \asymp D$, when $\beta_\sigma(D) = \eta_\sigma(C) + 1$ or $\beta_\sigma(C) = \eta_\sigma(D) + 1$.

A particular kind of Q-node that we will be using a lot is a *sequence of sets*. That is, a Q-node whose children are leaf P-nodes: P-nodes that are finite sets of symbols. We will

call such Q-nodes *simple* Q-nodes, and name them with letters $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{D}$. As Q-nodes, a set sequence and its reverse are equivalent.

Clearly a sequence of sets $\mathcal{A} = [A_1, \ldots, A_r]$ is a superstring collection for a hypergraph $H = (X, \mathcal{S} = \{S_1, \ldots, S_s\})$ if for each i, $1 \leq i \leq s$ there are j_i, k_i, $1 \leq j_i \leq k_i \leq n$ such that $S_i = \bigcup_{j_i \leq l \leq k_i} A_l$, and A_l, A_m are disjoint for all $j_i \leq l < m \leq k_i$. Given the hypergraph node S_i, we will denote by $\mathsf{J}_{\mathcal{A}}(S_i), \mathsf{K}_{\mathcal{A}}(S_i)$ the endpoints of S_i in $\mathcal{A}$, j_i, k_i respectively.

Denote a superstring of H, expressed by $\mathcal{A}$, by $\sigma(\mathcal{A})$. The size of $\mathcal{A}$ is defined to be the number of sets r, and denoted by $|\mathcal{A}|$. Denote by $\prec_{\mathcal{A}}$, $\lhd_{\mathcal{A}}$, and $\bowtie_{\mathcal{A}}$ the relations $\prec_{\sigma(\mathcal{A})}$, $\lhd_{\sigma(\mathcal{A})}$, and $\bowtie_{\sigma(\mathcal{A})}$, respectively. It follows that these relations are well-defined. Denote by $\overleftarrow{\mathcal{A}}$ the reverse sequence $[A_r, \ldots, A_1]$. Clearly $\overleftarrow{\mathcal{A}}$ is a superstring collection for $\mathcal{S}$ whenever $\mathcal{A}$ is, because $\mathcal{A}$ and $\overleftarrow{\mathcal{A}}$ are the two possible directions of the same Q-node. We will denote by $\mathcal{A}_{i,j}$ where $i \leq j$ the set $A_i \cup \cdots \cup A_j$. We will denote by $\mathcal{A}_{[i,j]}$ the subsequence $[A_i, \ldots, A_j]$.

4.1 The Merge *Operation on* Simple Q-nodes

We want to define the operation *merge*, $\mathbb{M}(\cdot, \cdot)$ between two simple Q-nodes. Intuitively $\mathbb{M}(\cdot, \cdot)$ should take as an input two set sequences $\mathcal{A}, \mathcal{B}$ that are superstring collections of the disjoint set collections $\mathcal{S}, \mathcal{T}$, respectively, and produce a superstring collection $\mathcal{D}$ for $\mathcal{S} \cup \mathcal{T}$. We want $\mathcal{D}$ to be as short as possible, therefore $\mathbb{M}(\cdot, \cdot)$ should find any "obvious" overlap between the sequences $\mathcal{A}, \mathcal{B}$. We will assume that $\mathcal{S} \cup \mathcal{T}$ is always a Sperner system, and that $\mathcal{S} \cap \mathcal{T} = \emptyset$. If $\mathcal{A}, \mathcal{B}$ are sequences of one element A, B respectively, it is easy to compute the "best" $\mathcal{D}$: $\mathcal{D} = [A \backslash B, A \cap B, B \backslash A]$. The size of the overlap is then $|A \cap B|$. How about if $\mathcal{A} = [A]$ is a singleton sequence but $\mathcal{B} = [B_1, B_2, \ldots, B_k]$, $k > 1$? Now the direction of the merge matters. We find the largest l such that $\mathcal{B}_{1,l} \subseteq A$ and $B_1, \ldots, B_l$ are pairwise disjoint. We can achieve overlap $|\mathcal{B}_{1,l}| + |(A \backslash \mathcal{B}_{1,l}) \cap B_{l+1}|$. We do the same with $\mathcal{A}$, $\overleftarrow{\mathcal{B}}$. Say w.l.o.g. that the largest overlap of the two is between $\mathcal{A}, \mathcal{B}$. Then $\mathcal{D} = [A \backslash \mathcal{B}_{1,l+1}, B_1, \ldots, B_l, A \cap B_{l+1}, B_{l+1} \backslash A, B_{l+2}, \ldots, B_k]$. It is technical, but not deep, to see how all this generalizes when $|\mathcal{A}| > 1$ and $|\mathcal{B}| > 1$:

4.1.1 Description

Algorithm *MERGE* : Given two set sequences $\mathcal{A} = [A_1, \ldots, A_p]$, $\mathcal{B} = [B_1, \ldots, B_q]$ that are superstring collections $\mathcal{A} = [A_1, \ldots, A_p]$, $\mathcal{B} = [B_1, \ldots, B_q]$ of disjoint set collections $\mathcal{S} = \{S_1^{\mathcal{A}}, \ldots, S_s^{\mathcal{A}}\}$ and $\mathcal{T} = \{T_1^{\mathcal{B}}, \ldots, T_t^{\mathcal{B}}\}$ respectively, do the following:[2]

1. For possible overlap size $l = 1$ upto $|S_s^{\mathcal{A}} \cap T_1^{\mathcal{B}}|$.

2. Find $l_{\mathcal{A}}, l_{\mathcal{B}}$ such that $|\mathcal{A}_{p-l_{\mathcal{A}}-1,p}| \geq l > |\mathcal{A}_{p-(l_{\mathcal{A}}),p}|$ and $|\mathcal{B}_{1,l_{\mathcal{B}}}| < l \leq |\mathcal{B}_{1,l_{\mathcal{B}}+1}|$. If $l \leq |A_p|$ then let $l_{\mathcal{A}} = -1$, and similarly if $l \leq |B_1|$ then let $l_{\mathcal{B}} = -1$.

3. Consider the two sequences of set sizes:

$$
\begin{aligned}
l_{\mathcal{A}} \geq 0 &\implies \mathsf{A} = [l - |\mathcal{A}_{p-l_{\mathcal{A}},p}|, |A_{p-l_{\mathcal{A}}}|, \ldots, |A_p|] =_{def} [a_1, \ldots, a_{l_{\mathcal{A}}+2}] \\
l_{\mathcal{A}} = -1 &\implies \mathsf{A} = [l] \\
l_{\mathcal{B}} \geq 0 &\implies \mathsf{B} = [|B_1|, \ldots, |B_{l_{\mathcal{B}}}|, l - |\mathcal{B}_{1,l_{\mathcal{B}}}|] =_{def} [b_1, \ldots, b_{l_{\mathcal{B}}+1}] \\
l_{\mathcal{B}} = -1 &\implies \mathsf{B} = [l]
\end{aligned}
$$

   Clearly $\sum_{i=1}^{l_{\mathcal{A}}+2} a_i = \sum_{i=1}^{l_{\mathcal{B}}+1} b_i = l$. Let $\mathsf{E} = [e_1, \ldots e_\alpha]$, $\alpha \leq l_{\mathcal{A}} + l_{\mathcal{B}} + 1$ be the "merging" of A, B. That is, cut the interval $[1, \ldots, l]$ on all places where A and B cut it, i.e. on $a_1, a_1 + a_2, \ldots, a_1 + \cdots + a_{p-1}, b_1, b_1 + b_2, \ldots, b_1 + \cdots + b_{q-1}$, resulting in a partition of $\{1, \ldots, l\}$, $\mathsf{E} = [e_1, \ldots, e_r]$, $r \leq p + q - 1$, $\sum_{i=1}^{r} e_i = l$.

4. Each e_i in E corresponds to some $a_{j'}$ in A and $b_{k'}$ in B. $a_{j'}$ in A and $b_{k'}$ in B in turn, correspond to some A_j in $\mathcal{A}$ and B_k in $\mathcal{B}$. For all such triplets (i, j, k) check that $|E_i = A_j \cap B_k| = e_i$.

5. Announce failure for overlap l, and repeat (2)-(5) with $l \leftarrow l - 1$, if for any triplet (i, j, k) the above condition fails.

[2] Assume that $\mathcal{S}$ and $\mathcal{T}$ are ordered by $\prec_{\mathcal{A}}$ and $\prec_{\mathcal{B}}$, respectively.

6. Else $\mathcal{A}, B$ can be merged with overlap l. The resulting sequence is

$$\mathcal{D} = [D_1, \ldots, D_r] = [A_1, \ldots, A_{p-l_{\mathcal{A}}-2}, A_{p-l_{\mathcal{A}}-1} \backslash \mathcal{B}_{1,l_{\mathcal{B}}+1}] \circ [E_1, \ldots, E_\alpha] \circ [B_{l_{\mathcal{B}}+1} \backslash \mathcal{A}_{p-l_{\mathcal{A}}-1,p}, B_{l_{\mathcal{B}}+2}, \ldots, B_q]$$

7. Repeat (1)-(6) for the pairs $(\mathcal{A}, \overleftarrow{\mathcal{B}}), (\overleftarrow{\mathcal{A}}, \mathcal{B}), (\overleftarrow{\mathcal{A}}, \overleftarrow{\mathcal{B}})$. Return the set sequence $\mathcal{D}$ resulting from the merging with the greatest overlap l, together with the overlap magnitude l.

Notice that we do not merge superstring collections with 0 overlap. We will denote by $\mathbb{O}(\mathcal{A}, \mathcal{B})$ the overlap associated with the merge $\mathbb{M}(\mathcal{A}, \mathcal{B})$. Notice that $\mathbb{O}(\cdot, \cdot)$ is symmetric. When merging $\mathcal{A}, \mathcal{B}$ in that order, with associated set collections $S_1^{\mathcal{A}} \bowtie_{\mathcal{A}} \cdots \bowtie_{\mathcal{A}} S_s^{\mathcal{A}}$, and $S_1^{\mathcal{B}} \bowtie_{\mathcal{B}} \cdots \bowtie_{\mathcal{B}} S_t^{\mathcal{B}}$ respectively, we say that the merge *introduces* $S_s^{\mathcal{A}} \bowtie S_1^{\mathcal{B}}$.

4.1.2 Correctness of MERGE

Theorem 3. (Correctness of MERGE): Let $\mathcal{A} = [A_1, \ldots, A_p]$, $\mathcal{B} = [B_1, \ldots, B_q]$, be superstring collections of $\mathcal{S} = \{S_1^{\mathcal{A}}, \ldots, S_s^{\mathcal{A}}\}$ and $\mathcal{T} = \{T_1^{\mathcal{B}}, \ldots, T_t^{\mathcal{B}}\}$ respectively. Let $\mathcal{S} \cup \mathcal{T}$ be a Sperner system and $\mathcal{S} \cap \mathcal{T} = \emptyset$. Then $\mathcal{D} = [D_1, \ldots, D_r] = \mathbb{M}(\mathcal{A}, \mathcal{B})$ is a superstring collection for $\mathcal{S} \cup \mathcal{T}$.

Proof. There are three ways to merge $\mathcal{A}, B$ depending on their sizes:

- *Case 1*: $|\mathcal{A}| = |\mathcal{B}| = 1$, $\mathcal{D} = [A_1 \backslash B_1, A_1 \cap B_1, B_1 \backslash A_1]$. Then w.l.o.g. $\mathcal{S} = \{S_1^{\mathcal{A}}\}$ and $\mathcal{T} = \{T_1^{\mathcal{B}}\}$. Clearly $\mathcal{D}$ is a superstring collection for $\mathcal{S} \cup \mathcal{T}$. For instance $S_1^{\mathcal{A}} = A_1 = (A_1 \backslash B_1) \cup (A_1 \cap B_1) = D_1 \cup D_2$.

- *Case 2*: $|\mathcal{A}| = 1$, $|\mathcal{B}| > 1$. Assume w.l.o.g. that $\mathcal{B}$ is not reversed before merging. Then for some l, $B_1 \cup \cdots \cup B_l \subset A_1$, $B_1, \ldots, B_l$ are pairwise disjoint, and $\mathcal{D} = [A \backslash \mathcal{B}_{1,l+1}, B_1, \ldots, B_l, A \cap B_{l+1}, B_{l+1} \backslash A, B_{l+2}, \ldots, B_k]$. Clearly $A = (A \backslash \mathcal{B}_{1,l+1}) \cup B_1 \cup \ldots \cup B_l \cup (A \cap B_{l+1})$, disjoint union. Therefore $S_1^{\mathcal{A}} = D_1 \cup \cdots \cup D_{l+2}$. Now for an arbitrary $T_i^{\mathcal{B}}$, let $j = \mathsf{J}_{\mathcal{B}}(T_i^{\mathcal{B}}), k = \mathsf{K}_{\mathcal{B}}(T_i^{\mathcal{B}})$. If $k \leq l$ then $T_i^{\mathcal{B}} = \mathcal{D}_{j+1,k+1}$. If $j \geq l+2$ then $T_i^{\mathcal{B}} = \mathcal{D}_{j+2,k+2}$. If $j \leq l+1 \leq k$ then $S_i^{\mathcal{B}} = \mathcal{D}_{j+1,k+2}$. Clearly the above are disjoint unions.

- *Case 3*: $|\mathcal{A}| > 1, |\mathcal{B}| > 1$. Assume w.l.o.g. that $\mathcal{A}, B$ are not reversed before merging. By symmetry, we only need to consider a set $S_i^{\mathcal{A}}$ expressed in $\mathcal{A}$. Let $j = \mathsf{J}_{\mathcal{A}}(S_i), k = \mathsf{K}_{\mathcal{A}}(S_i)$. Let $l_{\mathcal{A}}, l_{\mathcal{B}}$ and $\mathcal{D} = [D_1, \ldots, D_r]$ be as in the algorithm description. If $k \leq p - l_{\mathcal{A}} - 2$ then $S_i^{\mathcal{A}} = D_j \cup \cdots \cup D_k$. The case of sets $S_i^{\mathcal{A}}$ composed using some subsets from $A_{p-l_{\mathcal{A}}-1}, \ldots, A_p$ is a little trickier. Notice that the subsets $A_{p-l_{\mathcal{A}}-1}, \ldots, A_p \subseteq S_s^{\mathcal{A}}$ are part of the expression of $S_s^{\mathcal{A}}$ and therefore pairwise disjoint. Similarly the subsets $B_1, \ldots, B_{l_{\mathcal{B}}+1} \subseteq T_1^{\mathcal{B}}$ are pairwise disjoint. It follows then that $D_{p-l_{\mathcal{A}}-1}, \ldots, D_{p-l_{\mathcal{A}}+\alpha-1} = (A_{p-l_{\mathcal{A}}-1} \setminus B_{1,l_{\mathcal{B}}}), E_1, \ldots, E_\alpha$, are pairwise disjoint. Then we can express $S_i^{\mathcal{A}}$ as follows: let k' be the largest index among $1, \ldots, \alpha$ for the sets $E_1, \ldots, E_\alpha$, such that A_k appears on an intersection $E_{k'} = A_k \cap B_w$ for some $1 \leq w \leq \alpha$. If $j \leq p - l_{\mathcal{A}} - 2$ let $j' = j$. Else let j' be the smallest index among $1, \ldots, \alpha$ such that $E_{j'} = A_j \cap B_w$ for some $1 \leq w \leq \alpha$. Then $S_i^{\mathcal{A}} = D_{p-l_{\mathcal{A}}-1+j'} \cup \cdots \cup D_{p-l_{\mathcal{A}}-1+k'}$.

In all the above cases, the sets of $\mathcal{A}$ and $\mathcal{B}$ are expressed as disjoint unions of consecutive sets in $\mathcal{D}$. Therefore $\mathcal{D}$ is a superstring collection for $\mathcal{S} \cap \mathcal{T}$. □

Lemma 2. (*Optimality of MERGE*) Let $H = (X, \mathcal{S})$ be a hypergraph that has a non-repeating superstring. Let $\mathcal{A} = [A_1, \ldots, A_p], \mathcal{B} = [B_1, \ldots, B_q]$ be set sequences, let $A_1, \ldots, A_p$ be pairwise disjoint, and similarly let $B_1, \ldots, B_q$ be pairwise disjoint. Let $\mathcal{A}, \mathcal{B}$ be superstring collections for disjoint subsets of $\mathcal{S}$. Let $\mathbb{M}(\mathcal{A}, \mathcal{B}) = \mathcal{D} = [D_1, \ldots, D_r]$. For any non-repeating superstring σ of H, if (1) $\forall 1 \leq i \leq p$, A_i is expressed by σ, and $\forall 1 \leq i \leq q-1$, B_i is expressed by σ; (2) $\forall 1 \leq i \leq p-1, A_i \asymp_\sigma A_{i+1}$, and $\forall 1 \leq i \leq q-1, B_i \asymp_\sigma B_{i+1}$, then it is true that:

1. $\forall 1 \leq i \leq r-1, D_i \in \sigma$
2. $\forall 1 \leq i \leq r-1, D_i \asymp_\sigma D_{i+1}$

Proof. 1. Take an arbitrary D_i, in the sequence $\mathcal{D}$. If $D_i = A_j$ or $D_i = B_j$ for some j, it is immediate that $D_i \in \sigma$. If $D_i = A_j \cap B_k$ for some j, k, then since $A_j \in \sigma, B_k \in \sigma$, and σ is non-repeating, it follows that $A_j \cap B_k \in \sigma$.

2. Take arbitrary D_i, D_{i+1} in $\mathcal{D}$.

(a) If $D_i = A_j$ and $D_{i+1} = A_{j+1}$, or $D_i = B_j$ and $D_{i+1} = B_{j+1}$ for some j, then it is immediate that $D_i \asymp_\sigma D_{i+1}$.

(b) If $D_i = A_{p-l_\mathcal{A}-1} \backslash \mathcal{B}_{1,l_\mathcal{B}+1}$ then $D_{i+1} = A_{p-l_\mathcal{A}-1} \cap B_1$. Clearly σ expresses $\mathcal{B}_{1,l_\mathcal{B}+1}$ and since (1) σ expresses $A_{p-l_\mathcal{A}-1}$, (2) σ is non-repeating, and (3) $\mathcal{B}_{1,l_\mathcal{B}+1} \not\subseteq A_{p-l_\mathcal{A}-1}$, it follows that $A_{p-l_\mathcal{A}-1} \setminus \mathcal{B}_{1,l_\mathcal{B}+1} \asymp_\sigma A_{p-l_\mathcal{A}-1} \cap \mathcal{B}_{1,l_\mathcal{B}+1} \asymp_\sigma \mathcal{B}_{1,l_\mathcal{B}+1} \setminus A_{p-l_\mathcal{A}-1}$. Clearly then $D_i = A_{p-l_\mathcal{A}-1} \backslash \mathcal{B}_{1,l_\mathcal{B}+1} \asymp_\sigma A_{p-l_\mathcal{A}-1} \cap B_1 = D_{i+1}$.

(c) Having proven cases (a) and (b) let $D_{i-1} \asymp_\sigma D_i$ and let $D_i = A_j \cap B_k$. Consider the following cases:

i. $D_{i+1} = A_j \cap B_{k+1}$. We assume that $i > 1$ and $D_{i'} \asymp D_{i'+1}$ for any $i' < i$. Then $A_j \cap B_k \asymp_\sigma A_j \cap B_{k+1}$ follows from $A_j \in \sigma$, $B_k \asymp_\sigma B_{k+1}$, and the fact that σ is non-repeating.

ii. $D_{i+1} = A_{j+1} \cap B_k$. Similarly, since $B_k \in \sigma$ and $A_j \asymp_\sigma A_{j+1}$, it follows that $A_j \cap B_k \asymp_\sigma A_{j+1} \cap B_k$.

iii. $D_{i+1} = A_{j+1} \cap B_{k+1}$. Clearly either $A_{j+1} \subseteq B_{k+1}$ or $B_{k+1} \subseteq A_{j+1}$. Assume the former case, i.e. $A_{j+1} \subseteq B_{k+1}$ whereby $D_{i+1} = A_{j+1}$. Knowing that $D_{i'} \asymp_\sigma D_{i'+1}$ for $i' < i$, $A_j \cap B_k \asymp_\sigma A_{j+1}$ follows from the fact that $A_j \asymp_\sigma A_{j+1}$ and σ is non-repeating. Similarly in the latter case.

□

Lemma 3. Given two proper simple Q-nodes $\mathcal{A} = [A_1, \ldots, A_p]$ and $\mathcal{B} = [B_1 \ldots B_q]$, $\mathbb{M}(\mathcal{A}, \mathcal{B})$ is proper.

Proof. This is clear. In fact, referring to the description of MERGE, $\mathsf{L}(\mathbb{M}(\mathcal{A}, \mathcal{B})) = \sum_{i=1}^{r} |D_i|$.

□

We give an example of applying *Merge-Sequence-Pair* to two simple set sequences:

Example 1: Let

- $\mathcal{S} = \{S_1, S_2, S_3, S_4\}$

- $S_1 = \{a, b, c\}$; $S_2 = \{b, c, d, e\}$; $S_3 = \{a, c, d, e\}$; $S_4 = \{a, b, e\}$
- $\mathcal{A} = [A_1, A_2, A_3] = [\{a\}, \{b, c\}, \{d, e\}]$; $\mathcal{B} = [B_1, B_2, B_3] = [\{c, d\}, \{a, e\}, \{b\}]$

Therefore,

$$S_1 = A_1 \cup A_2, S_2 = A_2 \cup A_3, S_3 = B_1 \cup B_2, S_4 = B_2 \cup B_3$$

Then it is tedious to check that following the steps of the above algorithm results in

$$\mathbb{M}(\mathcal{A}, \mathcal{B}) = [\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{a\}, \{b\}]$$

with overlap = 3.

4.2 The GREEDY-MERGE-SPERNER Algorithm for Sperner Hypergraphs

Below we describe a greedy algorithm based on MERGE, restricted to Sperner hypergraphs (*GREEDY-MERGE-SPERNER*). The general algorithm will be shown in the next section*.

4.2.1 Description of GREEDY-MERGE-SPERNER

Algorithm *GREEDY-MERGE-SPERNER*: Let $H = (X, \mathcal{S} = \{S_1, \ldots, S_s\})$ be a Sperner hypergraph. A PQ-tree is built by incrementally constructing a P-node $P_t = \{\mathcal{A}_1, \ldots, \mathcal{A}_{r_t}\}$ where $\mathcal{A}_1, \ldots, \mathcal{A}_{r_t}$ are *simple* Q-nodes.

- For initialization set $t = 1$, $P_1 = \{[S_1], \ldots, [S_s]\}$.
- Do until $|P_t| = 1$ or for all pairs of sequences $\mathcal{A}, \mathcal{B} \in \mathcal{S}_t$, $\mathbb{O}(\mathcal{A}_t, \mathcal{B}_t) = 0$:
  1. Find $\mathcal{A}_t, \mathcal{B}_t$ in $\mathcal{S}_t$ such that $\mathbb{O}(\mathcal{A}_t, \mathcal{B}_t) > 0$ is maximum.
  2. Let $P_{t+1} = (P_t \backslash \{\mathcal{A}_t, \mathcal{B}_t\}) \cup \{\mathbb{M}(\mathcal{A}_t, \mathcal{B}_t)\}$.
  3. Set $t \leftarrow t + 1$.

Let $P_{GM/S} = \{\mathcal{A}_1, \ldots, \mathcal{A}_F\}$. Then $\mathcal{A}_i$ are Q-nodes and the set of solutions is the set of superstrings produced by the PQ-tree rooted at $P_{GM/S}$.

4.2.2 Properties of GREEDY-MERGE-SPERNER

Lemma 4. $P_{GM/S}$ is proper.

Proof. Immediate from the corresponding lemma for MERGE. □

Theorem 4. (Correctness of *GREEDY-MERGE-SPERNER*) Given a Sperner hypergraph H, the strings that $P_{GM/S}$ produces are superstrings for H.

Proof. The correctness of the above algorithm follows inductively from the correctness of MERGE. □

Theorem 5. Let H be a Sperner hypergraph that has non-repeating superstrings. Then *GREEDY-MERGE-SPERNER* produces a PQ-tree expressing all the shortest superstrings.

Proof. We first introduce a few concepts:

Definition. A non-repeating superstring σ for a Sperner hypergraph $H = (X, \mathcal{S} = \{S_1, \ldots, S_s\})$ such that $S_i \bowtie_\sigma S_j \implies S_i \cap S_j \neq \emptyset$ is called a *ladder*. We say that H is *flat* if there exists a ladder σ for H.

Any subcollection of sets of a Sperner hypergraph is a Sperner hypergraph. However, a subcollection H' of a flat Sperner hypergraph H need not be flat. If H' is flat, it is not hard to prove that a ladder σ for H contains a substring that is a ladder for H'. Intuitively a ladder corresponds to a simple Q-node that expresses the necessary and sufficient conditions for a superstring to be optimal. The following lemma expresses the above intuition:

Lemma 5. If σ is a ladder for the hypergraph H then the relation $\bowtie_\sigma$ restricted to elements of $\mathcal{S}$, is common to all optimal superstrings.

Proof. Assume w.l.o.g. $S_1 \lhd_\sigma S_2 \lhd_\sigma \ldots \lhd_\sigma S_s$. Since σ is non-repeating, for any $S_i, S_j \in \mathcal{S}$ with $i < j$, $|S_i \cap S_j| = max(0, \eta_\sigma(S_i) - \beta_\sigma(S_j))$. Since $\mathcal{S}$ is Sperner and for any i $|S_i \cap S_{i+1}| > 0$, it follows $|S_i \cap S_{i-1}| > |S_i \cap S_j|$ for any $j < i-1$, and $|S_i \cap S_{i+1}| > |S_i \cap S_j|$ for any $j > i+1$. Say for contradiction that in some optimal superstring σ' $S_i \bowtie S_x$ where $x \neq i-1, i+1$. Say

w.l.o.g. that $x > i$. From $|S_i \cap S_{i+1}| > |S_i \cap S_x|$ it follows that $S_x \bowtie_{\sigma'} S_i \bowtie_{\sigma'} S_{i+1}$. Say w.l.o.g. that $S_x \lhd_{\sigma'} S_i \lhd_{\sigma'} S_{i+1}$. Since $S_i \lhd_\sigma S_{i+1} \prec_\sigma S_x$ it follows that $|S_{i+1} \cap S_x| > |S_i \cap S_x|$, therefore $S_{i+1} \cap S_x \not\subseteq S_i \cap S_x$ and therefore σ' repeats symbols, contradicting its optimality. □

Lemma 6. Let σ be a ladder for $H = (X, \mathcal{S} = \{S_1 \bowtie_\sigma \cdots \bowtie_\sigma S_s\})$, $s \geq 2$. Let $\mathcal{A} = [A_1, \ldots, A_p] = \mathcal{A}(\sigma_{[1,i]}), \mathcal{B} = [B_1, \ldots, B_q] = \mathcal{A}(\sigma_{[i+1,s]})$, $1 \leq i < s$ be the associated superstring collections for $\{S_1, \ldots, S_i\}$, $\{S_{i+1}, \ldots, S_s\}$ respectively. Then $\mathbb{M}(\mathcal{A}, \mathcal{B})$ produces only non-repeating superstrings, including σ.

Proof. The algorithm will try the largest possible overlap $l = |S_i \cap S_{i+1}|$. It is not hard to see that, by virtue of the existence of σ yielding overlap l, condition 4 of the merging algorithm is satisfied for overlap l. □

Lemma 7. Let H be a flat Sperner hypergraph. Then every superstring produced by $P_{GM/S}$ is a ladder.

Proof. Let σ be a ladder for $\mathcal{S}$. Assume w.l.o.g. that $S_1 \bowtie_\sigma \ldots \bowtie_\sigma S_s$. It suffices to show inductively that at each step of *GREEDY-MERGE-SPERNER*, two simple Q-nodes $\mathcal{A}_t = [S_{\alpha_1}, \ldots, S_{\alpha_p}], \mathcal{B}_t = [S_{\beta_1}, \ldots, S_{\beta_q}]$ are merged in that order, to produce a non-repeating simple Q-node $\mathcal{D}_t$, and to introduce $S_{\alpha_p} \bowtie_\sigma S_{\beta_1}$. Say this is true up to step t. Since σ is a ladder and at step t the available simple Q-nodes are segments of σ by inductive assumption, there are merges of overlap > 0. Therefore $\mathbb{O}(\mathcal{A}_{t+1}, \mathcal{B}_{t+1}) > 0$. Then $S_{\alpha_p} \cap S_{\beta_1} \neq \emptyset$. Since σ is non-repeating, S_{α_p} has a neighbor $\neq S_{\alpha_{p-1}}$, in σ. Call this S_γ. Say w.l.o.g. that $S_{\alpha_{p-1}} \lhd_\sigma S_{\alpha_p} \lhd_\sigma S_\gamma$. By inductive assumption S_γ has at most one neighbor in $\mathcal{S}_t$. Say S_γ lies on $\mathcal{B}'$ in $\mathcal{S}_t$. Clearly $\mathbb{M}(\mathcal{A}_t, \mathcal{B}')$ can be performed to yield overlap $\mathbb{O}(\mathcal{A}_t, \mathcal{B}') = |S_{\alpha_p} \cap S_\gamma| > |S_{\alpha_p} \cap S_x|$ for any $S_{\alpha_p} \prec_\sigma S_x$. Therefore it must be that $S_{\beta_1} \prec_\sigma S_{\alpha_1} \lhd_\sigma \cdots \lhd_\sigma S_{\alpha_p}$. Since σ is non-repetitive $|S_{\beta_1} \cap S_{\alpha_1}| > |S_{\beta_1} \cap S_{\alpha_p}|$ Therefore $\mathbb{O}(\overleftarrow{\mathcal{A}}, \mathcal{B}) > \mathbb{O}(\mathcal{A}, \mathcal{B})$, a contradiction. □

A ladder σ for $\mathcal{S} = \{S_1, \ldots, S_s\}$ corresponds to a simple Q-node that is a non-repeating superstring collection for $\mathcal{S}$. To describe this Q-node, assume without loss of generality that $S_1 \lhd_\sigma \cdots \lhd_\sigma S_s$. Let $a_i = |S_i \backslash S_{i+1}|$ for $1 \leq i \leq s-1$. It can be readily seen that $\beta_i = \beta_\sigma(S_i) =$

$1 + \sum_{j=1}^{i-1} a_j$ and $\eta_i = \eta_\sigma(S_i) = -1 + |S_i| + \beta_\sigma(S_i)$. Any minimal string that satisfies these conditions, or the reverse of such a string, is a ladder for $\mathcal{S}$. All such superstrings are produced by the simple Q-node $\mathcal{A}(\sigma) = [\{\sigma_1, \ldots, \sigma_{e_1-1}\}, \ldots, \{\sigma_{e_t}, \ldots, \sigma_{|\sigma|}\}]$ where $[e_1, \ldots, e_t]$ is the ordered sequence of all numbers in the set $\{\beta_2, \ldots, \beta_s, \eta_1+1, \ldots, \eta_{s-1}+1\}$ each distinct number occuring only once in $[e_1, \ldots, e_t]$.

If an arbitrary Sperner hypergraph $H = (X, \mathcal{S})$ has non-repeating solutions, these solutions are permutations of a finite number of ladders. This is true because an arbitrary Sperner hypergraph can be partitioned in the obvious way into disjoint flat Sperner hypergraphs. That is $\mathcal{S} = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_k$, where each $\mathcal{S}_i$ is flat, with associated ladder σ_i. Then any $\sigma = v_1(\sigma_{\pi(1)}) \cdots v_k(\sigma_{\pi(k)})$ where π is a permutation on $\{1, \ldots, k\}$ and v_i is either the identity or the reversing operation, is an optimal, non-repeating superstring for H. These superstrings can be represented by a PQ-tree consisting of one P node on top, leading to k simple Q-nodes. Finally, it is clear that *GREEDY-MERGE-SPERNER* produces all such non-repeating superstrings and that $P_{GM/S}$ is a PQ-tree representation of the set of optimal solutions. □

4.3 Examples

In the following examples we demonstrate how merges selected and used to produce superstrings.

Example 2: Non-repeating solutions.

- $\mathcal{S} = \{S_1, S_2, S_3, S_4, S_5\}$
- $S_1 = \{a, b, c, d, e\}$; $S_2 = \{c, d, e, f\}$; $S_3 = \{f, g, h\}$; $S_4 = \{i, j\}$; $S_5 = \{j, k\}$

1. $P_1 = \{[S_1], \ldots, [S_5]\}$.
2. First merge is between sets of maximum intersection, in our case S_1, S_2. After that $P_2 = \{[\{\{a, b\}, \{c, d, e\}, \{f\}], [S_3], [S_4], [S_5]\}$
3. There are now two merges of overlap 1. Say the one between S_4 and S_5 is performed first. Then $P_3 = \{[\{\{a, b\}, \{c, d, e\}, \{f\}], [S_3], [\{i\}, \{j\}, \{k\}]\}$.

4. There is now one merge of overlap 1, namely between $[\{\{a,b\},\{c,d,e\},\{f\}]$ and $[S_3]$. After that is performed $P_4 = \{[\{\{a,b\},\{c,d,e\},\{f\},\{g,h\}],[\{i\},\{j\},\{k\}]\}$.

5. Now there is no more merge of nonzero overlap. The algorithm stops and returns the following solutions: let $\mathcal{A}_1 = [A_1, A_2, A_3, A_4] = [\{\{a,b\},\{c,d,e\},\{f\},\{g,h\}]$, $\mathcal{A}_2 = [A_5, A_6, A_7] = [\{i\},\{j\},\{k\}]$. Then $\sigma(\mathcal{A}_1) = \sigma(A_1)\ldots\sigma(A_4)$ where $\sigma(A_i)$ is a string listing the elements of A_i in any order. Similarly for $\sigma(\mathcal{A}_2)$. Furthermore, $\tau(\mathcal{A}_1)$ can be $\sigma(\mathcal{A}_1)$ or its reverse, and similarly for $\tau(\mathcal{A}_2)$. The algorithm returns any solution of the form $\tau(\mathcal{A}_1)\tau(\mathcal{A}_2)$ or $\tau(\mathcal{A}_2)\tau(\mathcal{A}_1)$. Equivalently, the algorithm returns the PQ-tree rooted at P_{FINAL} where P_{FINAL} has children the simple Q-nodes $\mathcal{A}_1, \mathcal{A}_2$.

In the following example there are many routes that *GREEDY-MERGE-SPERNER* can take. We just show one of them.

Example 3: Solutions with repeated probes.

- $\mathcal{S} = \{S_1, \ldots, S_6\}$
- $S_1 = \{a,b,c,d,e\}$; $S_2 = \{b,c,d,e,f\}$; $S_3 = \{a,c,d,f,g\}$; $S_4 = \{b,c,d,g,h\}$; $S_5 = \{a,b,f,g,h\}$; $S_6 = \{e,g\}$.

1. $P_1 = \{[S_1], \ldots, [S_7]\}$.
2. First merge is between S_1 and S_2: $P_2 = \{A = [\{a\},\{b,c,d,e\},\{f\}],[S_2],\ldots,[S_6]\}$.
3. Second merge is between S_4 and S_5: $P_3 = \{A,[S_3],B = [\{a,f\},\{b,g,h\},\{c,d\}],[S_6]\}$.
4. Third merge is between S_2 and B: $P_4 = \{A, D = [\{a,f\},\{b,h\},\{g\},\{c,d\},\{a,f\}],[S_6]\}$.
5. Fourth merge is between A and D:
   $P_5 = \{[\{a,f\},\{b,h\},\{g\},\{c,d\},\{f\},\{a\},\{b,c,d,e\},\{f\}],\ [S_6]\}$
6. There are no more merges to be performed. An example of a superstring returned is $\sigma =$ *af bh g cd f a b cde f ge*. Another example is $\sigma' =$ *eg fa hb g cd f a b dec f*. Spaces in these strings are just added to make clear the separation into sets in the set sequences. Notice that S_6 is in the "leftmost" place on σ' and in the "rightmost" place on σ.

4.4 The GREEDY-MERGE *Algorithm for General Hypergraphs*

Below we describe an extension of *GREEDY-MERGE-SPERNER*, the *GREEDY-MERGE* algorithm that retrieves the *C1P* (Booth and Lueker, 1976) in general hypergraphs. In particular *GREEDY-MERGE* will retrieve all optimal solutions whenever those are non-repeating.

4.4.1 Description of GREEDY-MERGE

Definition. A set B can be c-inserted in a simple Q-node $\mathcal{A} = [A_1, \ldots, A_k]$, where $\mathcal{A}$ is a superstring collection of hyperedge collection $\mathcal{S} = \{S_1, \ldots, S_s\}$ if (a) $B \subset S_i$ for some i, and (b) c is the smallest integer > 0 such that $B = (A_i \cap B) \cup A_{i+1} \cup \cdots \cup A_{i+c-1} \cup (A_{i+c} \cap B)$, or $B \subseteq A_i$ and $c = 0$. The resulting Q-node is $\mathcal{A}' = [A_1, \ldots, A_i \backslash B, A_i \cap B, A_{i+1}, \ldots, A_{i+c} \cap B, A_{i+c} \backslash B, A_{i+c+1}, \ldots, A_k]$.

The lemma below follows easily from the above definition:

Lemma 8. Let $\mathcal{A} = [A_1, \ldots, A_k]$ be a superstring collection for $H = (X, \mathcal{S} = \{S_1, \ldots, S_s\})$ and let B be c-inserted in $\mathcal{A}$ to yield $\mathcal{A}'$, with $c > 0$. Then $\mathcal{A}'$ is a superstring collection for $H' = (X, \mathcal{S} \cup \{B\})$.

It is trivial to prove an optimality lemma for c-insertions, similar to the one for the MERGE operation:

Lemma 9. Let $H = (X, \mathcal{S})$ be a hypergraph that has a non-repeating superstring. Let $\mathcal{A} = [A_1, \ldots, A_p]$ be a simple Q-node and let $\mathcal{A}' = [A'_1, \ldots, A'_{p'}] = [A_1, \ldots, A_i \backslash B, A_i \cap B, A_{i+1}, \ldots, A_{i+c-1}, A_{i+c} \cap B, A_{i+c} \backslash B, A_{i+c+1}, \ldots, A_p]$ be the result of c-inserting a set $B \in \mathcal{S}$ with $c > 0$ in $\mathcal{A}$. For any non-repeating superstring σ of H, if (1) $\forall 1 \leq i \leq p, A_i \in \sigma$; (2) $\forall 1 \leq i \leq p-1, A_i \asymp_\sigma A_{i+1}$, then it is true that:

1. $\forall 1 \leq i \leq r-1, A'_i \in \sigma$.
2. $\forall 1 \leq i \leq r-1, A'_i \asymp_\sigma A'_{i+1}$

Algorithm *GREEDY-MERGE* : Let $H = (X, \mathcal{S} = \{S_1, \ldots, S_s\})$ be a hypergraph. Define $\mathcal{L} = \{S_i \in \mathcal{S} | \forall l \neq i (S_l \in \mathcal{S} \implies S_i \not\subset S_l)\}$, a Sperner subgraph.

---

1. Construct top P-node $P_{GM}(H)$, initially *empty*.

2. Apply *GREEDY-MERGE-SPERNER* on $\mathcal{L}$ to create a PQ-tree for $\mathcal{L}$, $P_{GM/S}(\mathcal{L}) = \{\mathcal{A}_1, \ldots, \mathcal{A}_m\}$. Let all $\mathcal{A}_i$ be children to P_{GM}. Those may be modified in the subsequent steps.

3. Let $\mathcal{S}'$ be initialized to $\mathcal{S} \backslash \mathcal{L}$. Do:

   - Find a set $B \in \mathcal{S}'$ that can be c-inserted in some $\mathcal{A}_i$ with $c > 0$ and c-insert it. Remove B from $\mathcal{S}'$.

   until no set can be c-inserted with $c > 0$ in any $\mathcal{A}_i$. Call the resulting PQ-tree $P'_{GM/S}(\mathcal{L}) = \{\mathcal{B}_1, \ldots, \mathcal{B}_m\}$.

4. For each $\mathcal{B}_i$, for each $B_{ij} \in \mathcal{B}_i$, do:

   (a) Let $\mathcal{S}_{ij} = \{S_\alpha \in \mathcal{S}' | S_\alpha \subseteq B_{ij}\}$. Update $\mathcal{S}'$ to $\mathcal{S}' \backslash \mathcal{S}_{ij}$.

   (b) Run *GREEDY-MERGE* recursively on $H_{ij} = (\bigcup \mathcal{S}_{ij}, \mathcal{S}_{ij})$ to produce PQ-trees with P-nodes on top, $P^{ij}_{GM} = \{P^{ij}_1, \ldots, P^{ij}_{m_{ij}}\}$, where P^{ij}_k are P-nodes.

   (c) If *all* the resulting PQ-trees $P^{ij}_1, \ldots, P^{ij}_{m_{ij}}$ are non-repeating, then replace the P-node B_{ij} with the P-node $P_{GM}(B_{ij}) = \{P^{ij}_1, \ldots, P^{ij}_{m_{ij}}, elts(B_{ij} \backslash (\bigcup \mathcal{S}_{ij}))\}$ where $elts(B_{ij} \backslash (\bigcup \mathcal{S}_{ij}))$ are all the elements of $B_{ij} \backslash (\bigcup \mathcal{S}_{ij})$.

   (d) Else let *all* childern of P^{ij}_{GM} be children of $P_{GM}(H)$.

5. If $\mathcal{S}'$ is not empty, run *GREEDY-MERGE* on $\mathcal{S}'$ and let the children of the resulting P-node be childern of $P_{GM}(H)$.

4.4.2 Proof that GREEDY-MERGE *Retrieves the Consecutive Ones Property*

Theorem 6. $P_{GM}(H)$ is a superstring collection for H.

Proof. The proof is straight-forward. Here is a sketch:

- By correctness of *GREEDY-MERGE-SPERNER*, and by Lemma 9 follows that all the sets in $\mathcal{L}$ are expressed in any string produced by P_{GM}.

- The sets in $\mathcal{S}\backslash\mathcal{L}$ are of three categories:

  1. Sets that are c-inserted in step 3. It follows that these sets are expressed string produced by P_{GM}.
  2. Sets that are in $\mathcal{S}_{ij}$ of step 4. For those sets use the inductive assumption that P_{GM}^{ij} is a superstring collection for $\mathcal{S}_{ij}$. It easily follows that $P_{GM}(H)$ also expresses all sets in $\mathcal{S}_{ij}$ for any i, j.
  3. Sets that are dealt with in step 5 are expressed by superstrings of $P_{GM}(H)$ by inductive assumption.

Therefore all the sets in $\mathcal{S}$ are expressed in any superstring produced by P_{GM}. □

Theorem 7. If there exists a non-repeating superstring for H, $P_{GM}(H)$ produces all the optimal solutions.

Proof. Assume that hypergraph H has a non-repeating solution. It suffices to show that the tree rooted at $P_{GM}(H)$ returned by *GREEDY-MERGE* is proper, and produces all the non-repeating solutions.

First of all we argue that if H has non-repeating superstrings, then after step 4 of the algorithm $\mathcal{S}'$ is empty. Say that $S \in \mathcal{S}'$ after step 4. Clearly S has at least two elements. Since $S \notin \mathcal{L}$, $S \subseteq \bigcup \mathcal{B}_i$ for some i. Since S was not c-inserted, and $S \notin B_{ij}$ for any j, it follows S has two elements x, y s.t. $x \in B_{ij}, y \in B_{i(j+d)}$ with $d > 1$ and $S \neq (B_{ij} \cap S) \cup B_{i(j+1)} \cup \cdots \cup B_{i(j+d-1)} \cup (B_{i(j+d)} \cap S)$. It follows that under these conditions, any superstring σ that respects $B_{ij} \prec_\sigma B_{i(j+1)}$ and expresses S, has to repeat either x or y.

Consider any set $\mathcal{S}_{ij}$, defined in 4.(a) in the description of the algorithm. $\mathcal{S}_{ij}$ has non-repeating solutions, because H does. Inductively assume that P_{GM}^{ij} produces all non-repeating superstrings of $\mathcal{S}_{ij}$. Then it it follows that $P_{GM}(B_{ij})$ produces all and only the non-repeating superstrings of $\mathcal{S}_{ij} \cup \{B_{ij}\}$. Assuming that in a non-repeating superstring of $\mathcal{S}$ it is true that $B_{ij} \in \sigma$, $P_{GM}(B_{ij})$ produces all substrings expressing B_{ij}, that appear in any non-repeating superstring for $\mathcal{S}$.

Consider step 2 of *GREEDY-MERGE* . The nodes $\mathcal{A}_i$ are non-repeating superstring collections of the flat set collections $\mathcal{L}_i$ that partition $\mathcal{L}$. For any $\mathcal{A}_i = [A_{i1}, \ldots, A_{ir_i}]$ it follows by the optimality of MERGE, and by induction on the number of MERGE steps needed to construct $\mathcal{A}_i$ that any non-repeating superstring σ for $\mathcal{S}$ expresses $A_{ij}, 1 \leq j \leq r_i$. It similarly follows that for any non-repeating σ, $A_{ij} \asymp_\sigma A_{i(j+1)}$.

Consider now step 3 of *GREEDY-MERGE* . Each $\mathcal{B}_i$ is obtained from $\mathcal{A}_i$ by a series of c-insertions with $c > 0$. It follows by induction on the number of c-insertions that in any non-repeating superstring σ for $\mathcal{S}$, $B_{ij} \in \sigma$ and $B_{ij} \asymp_\sigma B_{i(j+1)}$.

Since H has non-repeating superstrings, $\mathcal{L}$ also has non-repeating superstrings. Therefore from the properties of *GREEDY-MERGE-SPERNER* step 2 of *GREEDY-MERGE* will return a proper tree rooted at $P_{GM/S}(\mathcal{L})$ producing all non-repeating superstrings of $\mathcal{L}$. Those are a superset of all the non-repeating superstrings of H. Step 3 of *GREEDY-MERGE* will update the tree to $P'_{GM/S}$ which produces all superstrings σ of $P_{GM/S}(\mathcal{L})$ that express all B_{ij} and satisfy all conditions $B_{ij} \asymp_\sigma B_{i(j+1)}$. Since those conditions are satisfied by any non-repeating superstring of H, $P'_{GM/S}$ also produces all such superstrings. Finally, step 4 of the algorithm updates each P-node B_{ij} to $P_{GM}(B_{ij})$ which as we showed by inductive assumption, produces all substrings expressing $\mathcal{S}_{ij} \cup \{B_{ij}\}$ of non-repeating superstrings of H. Therefore the algorithm will stop after step 4, and the final root P_{GM} produces all non-repeating superstrings of H.

It is clear that P_{GM} is proper. Therefore P_{GM} produces exactly the non-repeating superstrings of H, when at least one such superstring exists. □

Therefore *GREEDY-MERGE-SPERNER* and *GREEDY-MERGE* retrieve the optimal solution when this solution is non-repeating. This case corresponds to a physical mapping problem where there are no repeated probes. It also follows that the above algorithms retrieve the consecutive ones property. Therefore, *GREEDY-MERGE* is a generalization of algorithms that retrieve the *C1P* for general hypergraphs.

4.5 Approximation Guarantees

The *MAXSNP*-completeness theorem above shows that the problem is hard to approximate even when restricted to strict Sperner hypergraphs. We show below a version of *GREEDY-MERGE-SPERNER* that has a good approximation guarantee on strict Sperner hypergraphs.

4.5.1 The Algorithm 2-PHASE-GREEDY *for Strict Sperner Hypergraphs*

Let $H = (X, \mathcal{S} = \{S_1, \ldots, S_s\})$ be a strict Sperner hypergraph. Let π be an optimal ordering of the hyperedges derived from the optimal superstring σ_{OPT}. Then let $S_{\pi(1)}, \ldots, S_{\pi(s)}$ be the sets ordered optimaly, and let $q_i = \eta_{\sigma_{OPT}}(S_{\pi(i)}) - \beta_{\sigma_{OPT}}(S_{\pi(i+1)})$ be the optimal overlap between sets $S_{\pi(i)}$ and $S_{\pi(i+1)}$ according to σ_{OPT}. Then we can calculate the total optimal overlap, $\mathsf{Q} = \sum_{i=1}^{s-1} q_i$ and notice that $|\sigma_{OPT}| = -\mathsf{Q} + \sum_{i=1}^{s} |S_i| = -\mathsf{Q} + \mathsf{C}$.

Consider the following algorithm: First perform $\lceil \frac{(s-1)}{4} \rceil$ merges of maximal overlap between pairs of single sets in $\mathcal{S}$. Denote by o_i the overlap associated with the ith MERGE performed on H. It is trivial to see that $o_1 \geq q_i$ for any $1 \leq i \leq s-1$. Say that the two hyperedges merged in this first step to yield o_1 are $S_{\pi(\alpha_1)}, S_{\pi(\alpha_2)}$. Then consider o_2. Clearly $o_2 \geq q_i$ for any $1 \leq i \leq s$, $i \notin \{\alpha_1 - 1, \alpha_1, \alpha_2 - 1, \alpha_2\}$. The reason is that all q_i overlaps except at most the four overlaps $\{q_{\alpha_1 - 1}, q_{\alpha_1}, q_{\alpha_2 - 1}, q_{\alpha_2}\}$ associated with the hyperedges merged in step 1, are "available" at step 2, therefore a MERGE with at least that much overlap is performed. Similarly for o_i for any $1 \leq i \leq \lceil \frac{(s-1)}{4} \rceil$. It follows that $\mathsf{O}_\alpha = \sum_{i=1}^{\lceil \frac{(s-1)}{4} \rceil} o_i \geq \lceil \frac{\mathsf{Q}}{4} \rceil$.

Denote by σ_1 a superstring produced by applying the above procedure on $\mathcal{S}$. Since $\mathcal{S}$ is a strict Sperner hypergraph, $|\sigma_1| \leq -\mathsf{O}_\alpha + \mathsf{C}$. It is easy to see that $\mathsf{Q} \leq \lfloor \frac{1}{2} \times (\mathsf{C} - s) \rfloor \leq \frac{1}{2} \times (\mathsf{C} - s)$. It follows that the worst possible approximation ratio $\frac{|\sigma_1|}{|\sigma_{OPT}|}$ is given when $\mathsf{O}_\alpha = \frac{\mathsf{Q}}{4}$:

$$\frac{|\sigma_1|}{|\sigma_{OPT}|} \leq \frac{\mathsf{C} - \frac{\frac{\mathsf{C}-s}{2}}{4}}{\mathsf{C} - \frac{\mathsf{C}-s}{2}} = \frac{7\mathsf{C} + s}{4\mathsf{C} + 4s}$$

This in turn is $\leq \frac{7}{4} = 1.75$.

After the merges between pairs are performed to yield overlap O_α, we allow the algorithm to perform merges between single hyperedges, or between pairs of hyperedges merged

previously, but never between any set sequence that "contains" more than two hyperedges from $\mathcal{S}$. Consider now two adjacent sets $S_{\pi(i)}, S_{\pi(i+1)}$ in the optimal solution, with overlap q_i. If these hyperedges were not merged together in the first phase of the algorithm, they can still be merged now even if one or both of them have been merged already to another set. Call then q_i' the new maximal overlap between such two hyperedges. Clearly $\mathsf{Q}' = \sum_{i=1}^{s-1} q_i' \geq \mathsf{Q} - \mathsf{O}_\alpha$. Likewise as before using the best $\lceil \frac{(s-1)}{4} \rceil$ merging operations between pairs of hyperedges in $\mathcal{S}$ or set sequences that resulted from the first phase, we can retrieve an overlap $\mathsf{O}_\beta \geq \lceil \frac{\mathsf{Q}'}{4} \rceil$.

Including this second phase of the algorithm, let again σ_2 a superstring produced on $H = (X, \mathcal{S})$. It follows now that $|\sigma_2| \leq -\mathsf{O}_\beta - \mathsf{O}_\alpha + \mathsf{C}$. The worst possible approximation ratio is given when $\mathsf{O}_\alpha = \frac{\mathsf{Q}}{4}$ and $\mathsf{O}_\beta = \frac{(\mathsf{Q}-\mathsf{O}_\alpha)}{4} = \frac{3\mathsf{Q}}{4}$:

$$\frac{|\sigma_2|}{|\sigma_{OPT}|} \leq \frac{\mathsf{C} - \frac{\mathsf{C}-s}{4} - \frac{3(\frac{\mathsf{C}-s}{2})}{16}}{\mathsf{C} - \frac{\mathsf{C}-s}{2}} = \frac{50\mathsf{C} + 7s}{32\mathsf{C} + 32s}$$

Now the approximation ratio is $\leq \frac{50}{32} = 1.5625$. Call this algorithm *2-PHASE-GREEDY.*

Algorithm *2-PHASE-GREEDY*: Let $H = (X, \mathcal{S} = \{S_1, \ldots, S_s\})$ be a strict Sperner hypergraph.

---

1. Perform in a greedy fashion $\lceil \frac{(s-1)}{4} \rceil$ MERGE operations between pairs of single hyperedges in $\mathcal{S}$. These merges result in a PQ-tree rooted at the P-node P_α containing (i) hyperedges in $\mathcal{S}$, and (ii) Q-nodes resulting from the above merges between pairs of hyperedges.

2. Perform in a greedy fashion $\lceil \frac{(s-1)}{4} \rceil$ MERGE operations between children of P_α. These merges result in a PQ-tree rooted at the P-node P_β.

3. Set $t = 1, P_1 = P_\beta$.

4. Do until $|P_t| = 1$ or for all pairs of sequences $\mathcal{A}, \mathcal{B} \in \mathcal{S}_t, \mathbb{O}(\mathcal{A}_t, \mathcal{B}_t) = 0$:

   (a) Find $\mathcal{A}_t, \mathcal{B}_t$ in $\mathcal{S}_t$ such that $\mathbb{O}(\mathcal{A}_t, \mathcal{B}_t) > 0$ is maximum.

   (b) Let $P_{t+1} = (P_t \backslash \{\mathcal{A}_t, \mathcal{B}_t\}) \cup \{\mathbb{M}(\mathcal{A}_t, \mathcal{B}_t)\}$.

(c) Set $t \leftarrow t + 1$.

Claim 1 *2-PHASE-GREEDY* has an approximation ratio ≤ 1.5625.

Claim 2 *2-PHASE-GREEDY* has the following approximation ratios on set systems of sets with size bounded by K:

- $\leq \frac{157}{128} \approxeq 1.227$ when $K = 3$;
- $\leq \frac{207}{160} \approxeq 1.294$ when $K = 4$;
- $\leq \frac{257}{192} \approxeq 1.339$ when $K = 5$;
- $\leq \frac{307}{224} \approxeq 1.371$ when $K = 6$;
- $\leq \frac{357}{256} \approxeq 1.395$ when $K = 7$;
- $\leq \frac{407}{288} \approxeq 1.413$ when $K = 8$;
- $\leq \frac{457}{320} \approxeq 1.428$ when $K = 9$;
- $\leq \frac{507}{352} \approxeq 1.440$ when $K = 10$.

In particular we have proved that *MIN-HYPERGRAPH-SUPERSTRING* is *MAXSNP* - complete for sets of size bounded by $K \geq 5$. For $K = 5$ the best approximation ratio of *MIN-HYPERGRAPH-SUPERSTRING* is therefore $1 < c^* \leq 1.339$. Notice that the case $K = 2$ is trivial and *GREEDY-MERGE-SPERNER* solves it optimally.

In fact, *2-PHASE-GREEDY* preserves the optimality property of *GREEDY-MERGE- SPERNER* and consequently can be used instead of *GREEDY-MERGE-SPERNER* as part of *GREEDY-MERGE* to retrieve the C1p in arbitrary hypergraphs.

4.5.2 The Algorithm 2-LAYER-GREEDY

Consider a set system $\mathcal{S} = \{S_1, \ldots, S_s\}$, and a superstring $\sigma = \sigma_1 \cdots \sigma_n$ for $\mathcal{S}$. We say that σ_i is *part of* S_j if σ_i lies in the part of the superstring where the first occurrence of C_j is found. We say that there is a *transition* at place $(i, i+1)$ of σ if σ_i is part of a different collection of sets

than σ_{i+1}. We can divide σ into substrings, or *regions* ρ_j so that $\sigma = \rho_1 \cdots \rho_k$ so that for each $\rho_j = \sigma_{j_1} \cdots \sigma_{j_{i_j}}$ it is true that there is no transition at $(i, i+1)$ if $j_1 \leq i \leq i+1 \leq j_{i_j}$, but there is always transition at $(i, i+1)$ if for some j, σ_i is the last character of ρ_j and σ_{i+1} is the first character of ρ_{j+1}. Define the *weight* of a region ρ_j, denoted by $w(\rho_j)$ or briefly w_j, to be the number of sets in $\mathcal{S}$ the characters in ρ_j are part of. Then, the following equation clearly holds:

$$\sum_{i=1}^{s} |S_i| = \sum_{j=1}^{k} w(\rho_j) \times |\rho_j| \tag{1}$$

Restricted 2-Layer Hypergraphs: From now on we will be considering only Sperner collections $\mathcal{S} = \mathcal{S}_\mathcal{C} \cup \mathcal{S}_\mathcal{D}$ that satisfy the following properties:

1. $\mathcal{S}_\mathcal{C} = \{C_1, \ldots, C_s\}$ and $\mathcal{S}_\mathcal{D} = \{D_1, \ldots, D_t\}$ are disjoint strict Sperner collections.
2. For any $C \in \mathcal{S}_\mathcal{C}$, and for any $D, D' \in \mathcal{S}_\mathcal{D}$, $C \cap D \cap D' = \emptyset$.
3. For any $C, C' \in \mathcal{S}_\mathcal{C}$ and $D \in \mathcal{S}_\mathcal{D}$, if C' is 2-included in C, D then $C \cap C' \cap D = \emptyset$.
4. For any $D \in \mathcal{S}_\mathcal{D}$, D is 2-included in at most one pair $C, C' \in \mathcal{S}_\mathcal{C}$ so that $D \cap C \cap C' \neq \emptyset$.

We will call this a *restricted 2-layer hypergraph* (R2LH). Consider any superstring σ for the R2LH $\mathcal{S} = \mathcal{S}_\mathcal{C} \cup \mathcal{S}_\mathcal{D}$. Divide σ into regions so that $\sigma = \rho_1 \cdots \rho_k$ (such partitioning into regions is clearly unique). We claim that for any region ρ_i, $w(\rho_i) = w_i \leq 3$. To see this, notice that since $\mathcal{S}_\mathcal{C}$ is a Sperner system, ρ_i can be part of at most two sets from $\mathcal{S}_\mathcal{C}$. Similarly ρ_i can be part of at most two sets from $\mathcal{S}_\mathcal{D}$. So, $w_i \leq 4$ and $w_i = 4$ if ρ_i is part of C, C', D, D' for some $C, C' \in \mathcal{S}_\mathcal{C}$ and $D, D' \in \mathcal{S}_\mathcal{D}$. This is not possible because for any $C \in \mathcal{S}_\mathcal{C}$ and $D, D' \in \mathcal{S}_\mathcal{D}$ we know that $C \cap D \cap D' = \emptyset$. Similarly it follows that if $w_i = 3$ then ρ_i is part of some $C, C' \in \mathcal{S}_\mathcal{C}$ and $D \in \mathcal{S}_\mathcal{D}$ such that D is 2-included in C, C' and $D \cap C \cap C' \neq \emptyset$. The above hold for the optimal superstring σ_{OPT}.

Below is a 2-approximation algorithm on hypergraphs that are R2LH.

Algorithm: Let $G = (V, E, W)$ be the following graph:

(a) $V = \{v_1, \ldots, v_s\}$, where v_i corresponds to C_i in $\mathcal{S}_\mathcal{C}$.

(b) $E = \{\{v_i, v_j\} | \exists D \in \mathcal{S}_\mathcal{D} : D$ is 2-included in $C_i, C_j\}$.

(c) $w_{i,j} = W(\{i,j\}) = |C_i \cap C_j \cap D|$ where D is the set in $\mathcal{S}_\mathcal{D}$, 2-included in C_i, C_j, that maximizes $|C_i \cap C_j \cap D|$.

It follows that the above weighted graph is well-defined. We extend it with 0-weight edges to obtain a complete graph.

Now we can find the maximum cycle cover of the above graph in polynomial time. This is true because we can find the minimum cycle cover of a graph in polynomial time and therefore we can reverse the sign of the weight of the edges and find the maximum cycle cover of the graph in polynomial time. Another way to see this is the following: let $w_{max} = max\{w_{i,j} | \{i,j\} \in E\}$, and let $w'_{i,j} = W'(\{i,j\}) = 1 + w_{max} - w_{i,j}$. Clearly the new weigth (under W') of any cycle cover C of the graph G is $n \times w_{max} - \sum_{\{i,j\} \in \mathsf{C}} w_{i,j}$. Therefore minimizing the weight of a cycle cover under W' corresponds to maximizing the weight of a cycle cover under W.

The maximum cycle cover C consists of cycles $\mathsf{C}_1, \ldots, \mathsf{C}_l$. To obtain a superstring σ_C from C, the algorithm will "open" each cycle in the minimum weight edge. Say that a cycle contains nodes $i_i, \ldots, i_c$, corresponding to sets $C_{i_1}, \ldots, C_{i_c} \in \mathcal{S}_\mathcal{C}$. Say that $D_{j_1}, \ldots, D_{j_c}$ are 2-included in $(C_{i_1}, C_{i_2}), (C_{i_2}, C_{i_3}), \ldots, (C_{i_c}, C_{i_1})$ respectively, and therefore the intersection sizes $|D_{j_q} \cap C_{i_q} \cap C_{i_q+1}|$ correspond to the weights of the edges in C. The cycle can contain some 0-weight edges, corresponding to pairs of sets C, C' that have no associated set D. We say in such cases that the associated set D is the empty set. Assume without loss of generality that the minimum such intersection is the one involving D_{j_c} and therefore the cycle is opened at C_{i_c}, C_{i_1}. Consider the set sequence $A = [C_{i_1} \setminus D_{j_1}, (C_{i_1} \cap D_{j_1}) \setminus C_{i_2}, C_{i_1} \cap D_{j_1} \cap C_{i_2}, (D_{j_1} \cap C_{i_2}) \setminus C_{i_1}, C_{i_2} \setminus (D_{j_1} \cup D_{j_2}), (C_{i_2} \cap D_{j_2}) \setminus C_{i_3}, \ldots, C_{i_c} \setminus (D_{i_{c-1}} \cup D_{i_c}), C_{i_c} \cap D_{i_c}, D_{i_c} \setminus C_{i_c}]$, where the empty sets are omitted. A is a superstring collection for the subset $\{C_{i_1}, \ldots, C_{i_c}, D_{j_1}, \ldots, D_{j_c}\}$ of $\mathcal{S}$. To see this, just observe that each of the sets in the subset is expressed by a consecutive run of sets in A, and that consecutive run consists of pairwise disjoint sets. To see this last point just recall that for any $C \in \mathcal{S}_\mathcal{C}$ and $D, D' \in \mathcal{S}_\mathcal{D}$, $C \cap D \cap D' = \emptyset$. We can open all the cycles likewise, and merge the resulting set sequences to obtain a superstring for $\mathcal{S}_\mathcal{C}$ and for a subset of

$\mathcal{S}_{\mathcal{D}}$. We extend it in a trivial way to obtain a superstring for $\mathcal{S}$.

For a superstring $\sigma = \rho_1, \ldots, \rho_k$, define the sets Π_1, Π_2, Π_3 as follows: $\Pi_i = \{\rho_j | w_j = i\}$. Denote by $|\bigcup \Pi_i|$ the quantity $\sum_{\rho_j \in \Pi_i} |\rho_j|$.[3] Then from Equation (1), we get:

$$\sum_{i=1}^{s} |C_i| = \sum_{i=1}^{3} i \times |\bigcup \Pi_i| \tag{2}$$

Also, if we denote by Π_3^{C} the particular set Π_3 induced by the superstring constructed by "opening" the cycle cover as described above, notice the following: the weight of the maximum-weight cycle cover is an upper bound on $|\bigcup \Pi_3|$ for any superstring with associated Π_3. It follows then that $|\bigcup \Pi_3^{\mathsf{C}}| \geq \frac{1}{2} \times |\bigcup \Pi_3^{OPT}|$, where Π_3^{OPT} is the set Π_3 associated with the optimal superstring.

From the equations:

$$|\bigcup \Pi_3^{\mathsf{C}}| \geq \frac{1}{2} \times |\bigcup \Pi_3^{OPT}| \tag{3}$$

and

$$|\sigma_{OPT}| = |\bigcup \Pi_1^{OPT}| + |\bigcup \Pi_2^{OPT}| + |\bigcup \Pi_3^{OPT}| \tag{4}$$

and

$$|\sigma_{\mathsf{C}}| = |\bigcup \Pi_1^{\mathsf{C}}| + |\bigcup \Pi_2^{\mathsf{C}}| + |\bigcup \Pi_3^{\mathsf{C}}| \tag{5}$$

and

$$|\bigcup \Pi_1^{\mathsf{C}}| + 2|\bigcup \Pi_2^{\mathsf{C}}| + 3|\bigcup \Pi_3^{\mathsf{C}}| = \sum_{i=1}^{s} |C_i| = |\bigcup \Pi_1^{OPT}| + 2|\bigcup \Pi_2^{OPT}| + 3|\bigcup \Pi_3^{OPT}| \tag{6}$$

[3] $|\rho_j|$ is the number of symbols σ_i that are in ρ_j, i.e. the length of ρ_j.

it follows that $|\sigma_C| \leq 2|\sigma_{OPT}|$.

5. Experimental Results

We implemented the *GREEDY-MERGE-SPERNER* algorithm and ran it on randomly generated data. The data were generated according to the Lander-Waterman model, where clones are intervals of length 1 distributed uniformly along the interval $[0, N]$.[4] The interval $[0, N]$ was divided in $1000N$ discrete positions and probes were distributed along $[0, N]$ according to a Poisson process, except that for each clone C, a probe p was allowed to occur only once. Any occurrences of p in C after the first, were discarded. This distribution is very similar to a pure Poisson distribution if, as in our case, the mean arriving time of a probe is much greater than the length of a clone, which is 1 in our case. The hypergraph that was given as input to *GREEDY-MERGE-SPERNER* consisted of all the maximal generated clones.

Table 1 displays some results of running the algorithm while varying N, the length of the interval where the clones are distributed; n, the number of clones used for generating the data; m, the number of probes used for generating the data; and λ for exponential distribution of the arriving time of probes. p is the average number of probes after generating the data, r_{avg} is the actual average number of repetitions of probes, approximately $= \lambda N$, and r_{max} is the average over all generated sequences, maximum number of repetitions of a single probe. L_0 is the average length of the generated sequences, and L_{GM} is the average length of the sequences or sequence fragments produced by *SPERNER-GREEDY-MERGE*. To facilitate presentation, the performance is presented in percentage of optimal that correspond to the ratio L_0/L_{GM}. That is, when we say that the performance is 95.9% like on the table below in the experiment running with $N = 20$ and 300 probes, we mean that *SPERNER-GREEDY-MERGE* produces on average a superstring collection of total length 1.0428 × [length of the initial sequence].

[4]The clone beginnings are distributed along $[0, N-1]$ with uniform probability.

N	n	m	p	r_{avg}	r_{max}	L_0	L_{GM}	Performance
5	200	200	159.2	1.6	3.9	259.1	292.7	88.7%
10	100	200	118.3	1.4	3.8	165	163.2	100%
10	100	200	145	1.5	3.7	216.5	238.8	90.7%
10	100	200	159	1.7	4.7	268	319.5	84.2%
20	100	200	186.7	2.4	6.8	451.3	453.8	99.5%
20	100	200	192.8	3	7.1	555.3	585.5	94.9%
20	100	200	196.4	3.4	7.8	660.3	699	94.5%
20	100	300	275.5	2.4	6.5	638	665.5	95.9%
30	100	300	293	3.3	8.5	951	913	100%
30	150	300	293	3.3	8.5	969	1041	93.1%
40	200	400	397.5	4.9	12.5	1886.5	1937.5	97.4%

Table 1. Performance of *GREEDY-MERGE-SPERNER* on simulated data.

As can be seen, the major factor that seems to hurt the performance of the algorithm is the *coverage* of the gene, i.e. the average number of clones that cover each point in the interval $[0, N]$. This means that a hypergraph that is Sperner decomposable in a few layers is easier to handle than one that is decomposable in many layers. This experimental observation is consistent with our intuition that the *minimal Sperner decomposition* problem captures the essence of the difficulty of computing minimal superstrings. High probe repetition also hurts the performance of the algorithm, as expected. Finally, the performance of the algorithm increases with the number of probes, and seems unaffected as the number of clones increases. Occasionally the algorithm produces a shorter superstring than the initial superstring. This would correspond to experimental conditions where either too few clones, or too few probes are used, resulting in under-specified instances of the problem.

6. Conclusion

Physical mapping experimental techniques involve the collection of data that establish overlap relationships on a given set of clones. The aim is to find the most likely layout of the clones, under certain assumptions on the characteristics of the clones (eg. length, density of coverage of the genomic region), and under given kinds and rates of experimental errors. Physical mapping has been a very important method for obtaining genomic sequence. Recent achievements of the technique include the sequencing of the nematode *C. elegans* (The C. elegans Sequencing Consortium, 1998) and of the human Chromosome 22 (Collins et al., 1995, Hunt et al., 1999). In both projects physical mapping of the sequenced region was used first, and subsequently clones were sent to the sequencing pipeline.

The data from physical mapping experiments are of a combinatorial nature. Different physical mapping techniques, and the various assumptions on types of errors, translate beautifully into challenging combinatorial/computational problems. Algorithms usually optimize some parsimony criterion. In our case, the length of the superstrings is minimized. Studying such computational problems is interesting on its own, and has the additional motivation that some of the results may be applicable to a real physical mapping experiment. We believe that the problems studied in this chapter are interesting from a computation theory perspective. Moreover, our results may be applicable if further work is done towards the generalization of the algorithms to cope with all kinds of errors that occur in practice, and towards elaborate implementation and testing of the algorithms on real data.

Chapter 2

SEQUENCING A GENOME BY WALKING WITH BAC-ENDS

1. Introduction

Various approaches that have been proposed for sequencing large genomes broadly fall into two categories:

(i) Whole-genome shotgun sequencing. Shotgun sequencing involves breaking a target into random fragments, sequencing these fragments and reconstructing the full sequence from these pieces. Shotgun sequencing was invented by Sanger, who applied it to the genome of *bacteriophage* λ (Sanger et al. 1980, 1982). It was subsequently extended to genomes of large recombinant plasmids, large viruses, mitochondria, chloroplasts and bacteria (Goebel et al. 1990; Oda et al. 1992; Ohyama et al. 1986; Fleischmann et al. 1995). More recently, Weber and Myers (1997) proposed that the approach could be extended to very large genomes, such as the human genome, by making greater use of long-range linking information — for example, sequences at the opposite ends of large-insert clones. There are many potential pitfalls, such as the possibility that the huge number of repeats (for example, one million copies of the Alu repeat in the human genome) may result in many large-scale sequence misassemblies that are hard to detect and correct (Green, 1997).

(ii) *Clone-based shotgun sequencing*. This approach involves obtaining a collection of large-insert clones covering a genome and performing shotgun sequencing on each clone. This approach was used for sequencing the genomes of the yeast *S. cerevisiae* (Oliver et al. 1992; Dujon, 1996) and *C. elegans* (The C. elegans Sequencing Consortium, 1998), and is being used for sequencing of mammalian genomes such as the human and mouse. Clone-based sequencing has the advantage that it eliminates the possibility of large-scale misassemblies, because each clone is assembled individually. But, it requires that one generate the collection of clones covering the genome. It also has the drawback that there is some redundant sequencing due to the overlap of adjacent clones. (There is no workable technique for shotgun sequencing only the non-

overlapping portion of a clone. Each clone must be subjected to full shotgun sequencing, with the overlapping portion therefore being shotgun-sequenced twice. Such duplication can be avoided in the directed finishing phase, in which gaps and ambiguities in the shotgun sequence are resolved. Such overlaps thus increase the cost for the region by somewhat less than two-fold.)

The current cloning system of choice for clone-based sequencing is the Bacterial Artificial Chromosome (BAC) with inserts of roughly 200 kilobases (kb). The discussion below will refer to BACs (rather than generic 'large-insert clones') but the results, of course, are completely general.

Two approaches have been proposed for generating the BAC clones to be sequenced:

(i) *Physical Mapping (or 'map first, sequence second')*. This approach involves constructing a complete physical map covering the genome before beginning sequencing ('map first, sequence second'). One 'fingerprints' a BAC library, using a technique such as restriction digestion to characterize each clone, and then attempts to use this information to infer the order of the clones. A 'path' of clones is then selected for sequencing. This approach was successfully used to generate physical maps of cosmids used to sequence the *S. cerevisiae* and *C. elegans* genomes (Olson et al. 1986; Coulson et al. 1986). It is being applied to the human genome, although the problem is more challenging due to the larger clones sizes (yielding more complex fingerprints), the larger genome size and the more complex repeat structure.

(ii) *Walking (or 'map as you go')*. This approach proceeds directly to sequencing without a prior physical map: One starts by sequencing an initial collection of random clones and then 'walks' the genome by iteratively selecting minimally overlapping clones. Venter et al. (1996) proposed an efficient method for selecting minimally overlapping clones, which is commonly referred to as 'BAC-end sequencing' or, more formally, as 'Sequence-Tagged Connectors' (STCs). A BAC library is characterized by sequencing the inserts of each clone at its two ends (using primers located in the two vector arms), and a database of the resulting BAC-end sequences is created. Given a fully sequenced BAC clone C, one can walk by searching the database to identify all overlapping BACs. The location and orientation of each overlapping BAC is immediately apparent from the position of its end-sequence within C (Figure 2.1). One can also tell whether an overlapping BAC closes a gap in the genomic sequence by whether its other end-sequence lies in another fully-sequenced BAC clone C'.

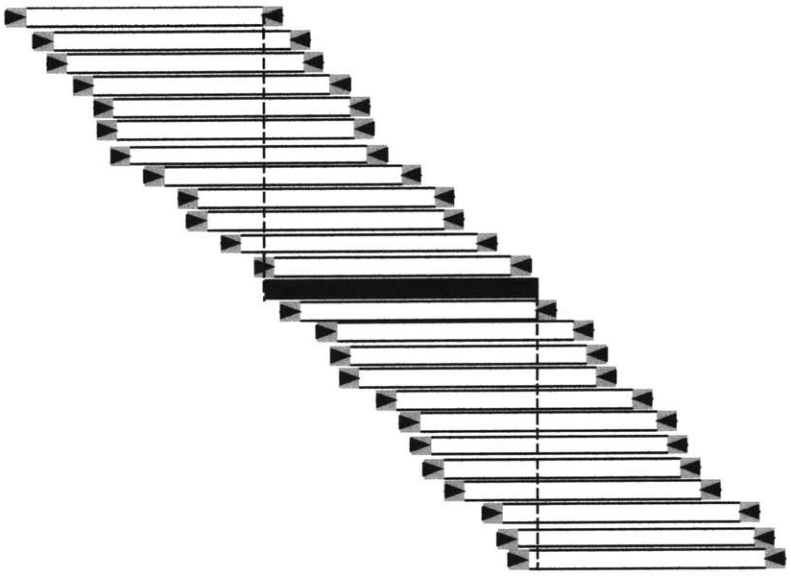

Figure 2.1. Overlapping BACs in library with depth d=12-fold. A typical BAC (shown in black) will tend to have 12 BACs overlapping its right end and 12 BACs overlapping its left end. Each overlapping clone can be identified because one of its end-sequences (shown by arrows) is contained within the sequence of the target BAC. The precise position of each overlapping clone is inferred by the location and orientation of the end-sequence within the sequence of the target BAC.

In this chapter we analyze the strategy of sequencing a genome by clone-based walking. Although the basic notion of walking is straightforward, key strategic questions have not been addressed: How many independent walks should be performed in parallel? With too few, one will need too many walking steps to cover the genome in a reasonable time. With too many, one may perform too much redundant sequencing as walks 'bump' into one another with random—and therefore suboptimal—overlap. How can one maximize the speed of covering the genome, while minimizing the amount of redundant sequencing?

We present a mathematical analysis that expresses the amount of redundant sequencing in terms of the initial density of walks and the depth of the BAC library. We also describe and analyze a variant strategy in which one employs a second BAC library with smaller-insert clones to close gaps efficiently; this strategy dramatically reduces the amount of redundant sequencing. The results provide direct guidance for planning a genome sequencing project.

2. Basic Model

We begin by describing the basic model to be analyzed.

BAC Library. We will initially consider the case of a single BAC library with the following properties:

(i) The clones have constant size L. The clone size L will serve as our unit of distance, so that we may set L = 1. (Later, we will consider the use of a second BAC library with inserts of constant size L' < 1.)

(ii) The clones are randomly segments of the genome. We thus ignore the possibility of cloning bias.

(iii) The library covers the genome to depth d — that is, the average number of clones covering a point in the genome is d.

(iv) The clones in the library have all been sequenced at both ends, yielding sufficient unique sequence to reliably detect overlap. We thus ignore the possibility that some BAC ends may not have been sequenced due to technical errors or may contain repeat sequences that make it impossible to recognize overlap.

Sequencing. We assume that the procedure for genomic sequencing satisfies the following conditions:

(i) The cost of sequencing a BAC is directly proportional to the length of its insert. This agrees with current practice in large-scale genomic sequencing centers, which perform a fixed number of shotgun sequences per kilobase of insert size. (A constant number of reads per kb may not be precisely optimal. On one hand, it has been suggested that smaller clones may require slightly fewer reads per kb to reach closure (Roach, 1995). On the other hand, smaller clones may require slightly more reads per kb of insert, because the cloning vector comprises a slightly larger proportion of the total clone length and thus of the reads from the shotgun library. In any case, these effects are small and offsetting.)

(ii) The cost of producing a small-insert shotgun library from a BAC (which involves preparing, shearing and cloning DNA) is negligible compared to the cost of performing the shotgun reads needed to sequence the BAC. This reflects current practice, for which the former is only about 2% of the latter.

In principle, one can sequence the genome with minimal overlap by sequencing a single initial seed clone C_0 and then walking successively outward by sequencing minimally overlapping clones, C_{-i} and C_i, on each side until one covers the entire genome (Figure 2.2). The resulting sequence of clones is commonly called a minimal tiling path. (Strictly speaking, it should be referred to as a minimal tiling path for the given library and choice of starting clone.) At each step, minimally overlapping clone C_i is identified by comparing the database of BAC-end sequences to the completed sequence of C_{i-1} to find the BAC-end sequence that lies closest to the growing end and points outward (Figure 2.1). We assume that there is always at least one overlapping clone pointing outward. This is a reasonable assumption, provided that the depth d is sufficiently large. For example, a BAC library with 200 kb inserts covering a mammalian genome of 3×10^9 bp to depth d=10 should yield only 6.8 gaps — ignoring chromosome ends (Lander and Waterman, 1988).

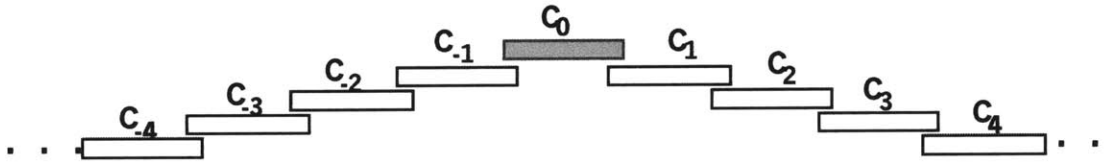

Figure 2.2. Serial walking of the genome from a single initial clone C_0.

It is straightforward to analyze the expected amount of redundant sequencing. On average, each successive clone overlaps the previous sequence by (1/d) of its length and provides (1-(1/d)) of new sequence. The ratio of redundant sequence to unique sequence is thus:

$$R = d^{-1}/(1-d^{-1}) = 1/(d-1).$$

A BAC library with depth d = 10 thus yields a minimal tiling path with redundant sequencing of 11.1%, while one with depth d= 20 entails redundant sequencing of 5.3%.

In reality, sequencing large genomes by serial walking is completely impractical owing to the cycle time to process each BAC clone: a mammalian genome would require 15,000 serial steps. Sequencing each BAC requires: growing the clone; preparing DNA; shearing the DNA; constructing a small-insert shotgun library; performing shotgun sequencing of clones from the small-insert library; assembling the reads by computer; and closing remaining gaps. Various quality assessment steps are performed along the way to ensure high yield. Although each individual step is straightforward, the overall elapsed time is currently on the order of 1-2 months.

The obvious solution is to process many BACs in parallel—that is, to simultaneously walk from many seed clones (Figure 2.3). Ideally, the seed clones should be dense enough that one could cover the vast majority of the genome in a modest number of walking steps. A reasonable goal would be to cover a mammalian genome in roughly one year.

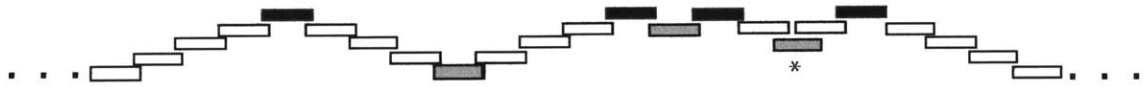

Figure 2.3. Serial walking of the genome from a collection of seed clones (shown in black). Bidirectional walking steps are shown, with steps that close an ocean shown in gray. The oceans are closed with relatively little overlap in the first two cases, but with large overlap in the third case (marked with asterisk).

Parallel walking, however, introduces a problem: the various walks may join with large overlaps, substantially increasing the proportion of redundant sequencing. Figure 2.3, for example, shows three clones (shown in gray) that join walks. The first two instances involve walks that fortuitously meet with relatively little overlap, but the third instance (marked with an asterisk) involves a walk that meets with large overlap and thereby result in substantial redundant sequencing.

It is important to understand how the expected proportion of redundant sequencing depends on the density of seed clones and the depth of the library. Section 3 presents the mathematical analysis. The reader interested primarily in applications may wish to proceed directly to Section 4.

3. Mathematical Analysis and Results

Our goal is to derive simple formulas providing a good approximation for the proportion of excess sequencing. Toward this end, we will make certain simplifying assumptions. The reasonableness of the approximations will then be demonstrated by their close agreement with simulations in Section 8.

4.1 Using one Library of Constant Size Clones

We start with a collection of non-overlapping seed clones. Following the terminology of Lander-Waterman (1988), each clone is an "island" followed by an "ocean" to be walked. Walking proceeds outward from each clone in a bidirectional fashion, with overlapping clones recognized on the basis of their end-sequence. At each round, one identifies instances in which two islands can be joined by a single clone (readily apparent from the two end-sequences) and ensures that a walking step is taken from only one of the two islands (to avoid unnecessary excess sequencing).

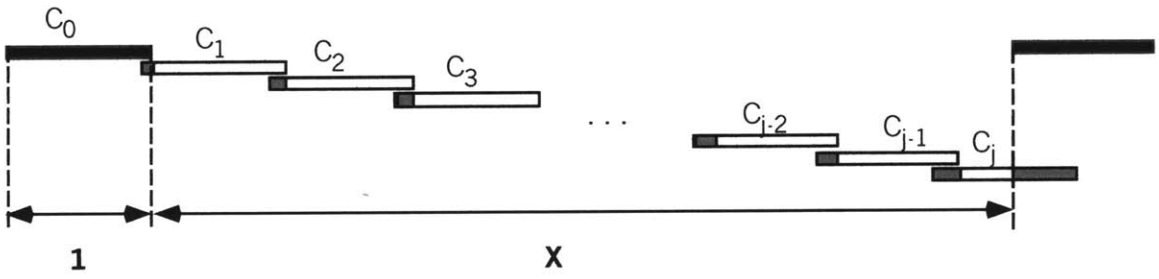

Figure 2.4. Unidirectional walking from a seed clone C_0. Each successive clone C_i extends the island, with the redundant sequence shown in gray. The last clone C_j closes the ocean, with redundant sequencing due to both overlap with C_{j-1} and with the next seed clone to the right. Each clone has size 1 and the ocean has size X. The number of clones sequenced is j+1, with each having length 1. The amount of unique sequence obtained (the initial island together with the ocean) is X+1.

Although walking proceeds bidirectionally in practice, it simplifies the discussion to suppose that walking proceeds unidirectionally to the right. Each bidirectional walking step is clearly equivalent to two unidirectional walking steps. Figure 2.4 shows a typical seed clone C_0 followed by consecutive walking steps $C_1, C_2, \ldots, C_j$, where C_j is the first clone that overlaps the seed clone to the right of C_0 (a fact that is readily apparent from its end sequence).

Suppose that the seeds cover proportion π of the genome in clones of length 1. The oceans therefore cover proportion $1-\pi$ and the mean ocean size must be $\omega = (1-\pi)/\pi$.

Our first simplifying assumption concerns ocean lengths:

Assumption of Exponential Oceans (AEO). Non-overlapping seed clones will be assumed to be chosen such that the resulting ocean lengths follow an exponential distribution (with mean ω).

What is the proportion of redundant sequencing entailed in sequencing the island C_0 together with the ocean on its right? Let the random variable J denote the sum of the lengths of $C_1, C_2, \ldots, C_j$. (Since the clones have unit length, J=j.) The amount of total sequencing is J+1, while the amount of unique sequence obtained is X+1 (see Figure 2.4). The expected proportion of redundant sequencing is thus:

$$R = \frac{(E(J)+1)}{(E(X)+1)} - 1,$$

where E( ) denotes the expected value. As noted above, the mean ocean size E(X) is $\omega = (1-\pi)/\pi$. Therefore, we simply need to calculate E(J).

Calculating E(J) precisely is complicated, but it is possible to make a good estimate by using a second simplifying assumption:

Assumption of Constant Overlap (ACO). Each clone C_i will be assumed to overlap the previous clone C_{i-1} by exactly the expected amount of overlap, 1/d, and thus to extend the sequence by exactly the expected amount, 1-(1/d).

ACO greatly simplifies the mathematical analysis, owing to an elegant property of exponential distributions: If each island is extended by a constant amount, then the lengths of the remaining oceans (that is, those that are not closed) follow the same exponential distribution as the initial oceans. This stability property allows the walking process to be analyzed by a simple recursion.

Proposition 1. Suppose that the genome is sequenced by seeding with non-overlapping clones to coverage π and then walking using a library with depth d. Let $\omega = (1-\pi)/\pi$. Assuming that the ocean sizes are exponentially distributed (AEO) and consecutive walking steps have constant overlap (ACO), we have:

(i) The expected proportion of redundant sequencing is

$$R(d,\omega) = \frac{E(J)+1}{\omega+1} - 1$$

(ii) $E(J) = 1/p_{d,\omega}$, where $p_{d,\omega} = 1 - e^{-(1-d^{-1})/\omega}$ is the probability that an ocean is closed with a single walking step.

(iii) The proportion of genome not sequenced after k unidirectional walking steps is $(1-\pi)(1-p_{d,\omega})^k = (1-\pi)\ e^{-k(1-d^{-1})/\omega}$. The proportion thus decreases geometrically with each walking step.

Proof. Part (i) was noted above. A formal proof follows from a straightforward application of Wald's equation (see Ross, 1970). Part (ii) employs ACO. The total clone length J required to close an ocean can be calculated by considering a single walking step. With probability $p_{d,\omega}$, the step closes the ocean, resulting in a total clone length of 1. With probability $1-p_{d,\omega}$, the step fails to close the ocean and leaves a remaining ocean having the same exponential distribution. It follows by recursion that

$$E(J) = (p_{d,\omega})\ 1 + (1- p_{d,\omega})\ (1 + E(J)).$$

The desired result follows by solving for E(J). Part (iii) follows by observing that the lengths of the remaining oceans after k walking steps continue follow the same exponential distribution. The total ocean length remaining after k steps is thus directly proportional to the proportion of oceans that remain unclosed after k steps. The proportion of the remaining oceans that are closed at each walking step is $p_{d,\omega}$ and thus the proportion remaining unclosed after k steps is $(1-p_{d,\omega})^k$. This completes the proof.

In fact, Proposition 1(i) can be generalized to any distribution of initial oceans sizes as follows.

Proposition 2. Suppose that the genome is seeded as in Proposition 1, but that the ocean sizes x have probability density f(x). Assuming ACO, the expected proportion of redundant sequencing is

$$R(d, \omega) = \frac{E(J)+1}{\omega+1} - 1,$$

where

$$E(J) = \int_0^\infty \lceil x/(1-d^{-1}) \rceil f(x)dx$$

and $\lceil x \rceil$ denotes the ceiling function (that is, the smallest integer $\geq$ x).

Proof. Under ACO, each walking step extends by distance $(1\text{-}d^{-1})$. The number J of walking steps needed to close an ocean of size x is thus $\lceil x/(1-d^{-1}) \rceil$. The result follows simply by taking the expectation of J over the distribution of ocean sizes.

We now apply the results to study the problem of sequencing a genome by walking. Figure 2.5 shows the proportion R of redundant sequencing, as a function of the initial mean ocean length ω and the BAC library depth d. If the genome is seeded to leave oceans of average length $\omega = 1$ (corresponding to initial coverage $\pi = 50\%$), the proportion of redundant sequencing ranges from 32% to 29% as the BAC library depth ranges from d=15 to $d = \infty$. If the genome is seeded more sparsely so that $\omega = 2$ and $\pi = 33\%$, the redundant sequencing R ranges from 23% to 18% over the same range of BAC library depth. If the oceans are still larger ($\omega = 3$ and $\pi = 25\%$), the redundant sequencing R ranges from 19% to 13%.

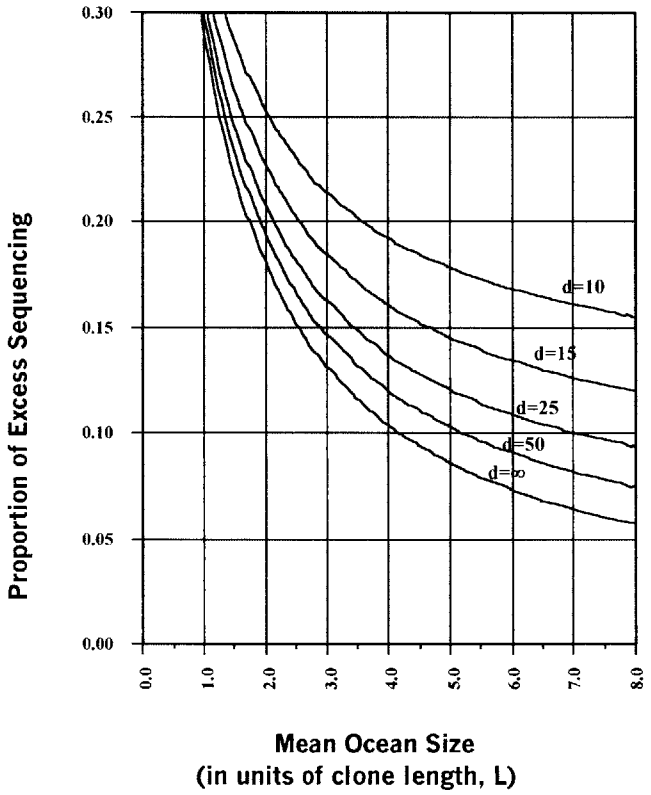

Figure 2.5. Proportion of excess sequencing. Curves show the proportion of excess sequencing, $R(d,\omega)$ as a function of mean initial ocean size ω for various library depths d = 10, 15, 25, 50 and ∞.

It is useful to compare the results to the situation of serially walking the entire genome with a minimal tiling path (which corresponds to ocean size $\omega = \infty$). As noted above, the proportion of redundant sequencing for a minimal tiling path is 1/(d-1). By subtracting this quantity from R, we find the additional redundant sequencing R* caused by inefficient closure of oceans. Figure 2.6 shows the corresponding graph for R*.

The graph shows that the redundant sequencing caused by inefficient closure of oceans depends primarily on the mean ocean size, but much less on the library depth d. This makes intuitive sense, because the inefficiency arises primarily from closing small oceans with clones of unit length. The availability of more clones in a deep library thus makes only a modest improvement. Indeed, the component R* is decreased by only a few percentage points by going from a library with d = 10 to d = ∞.

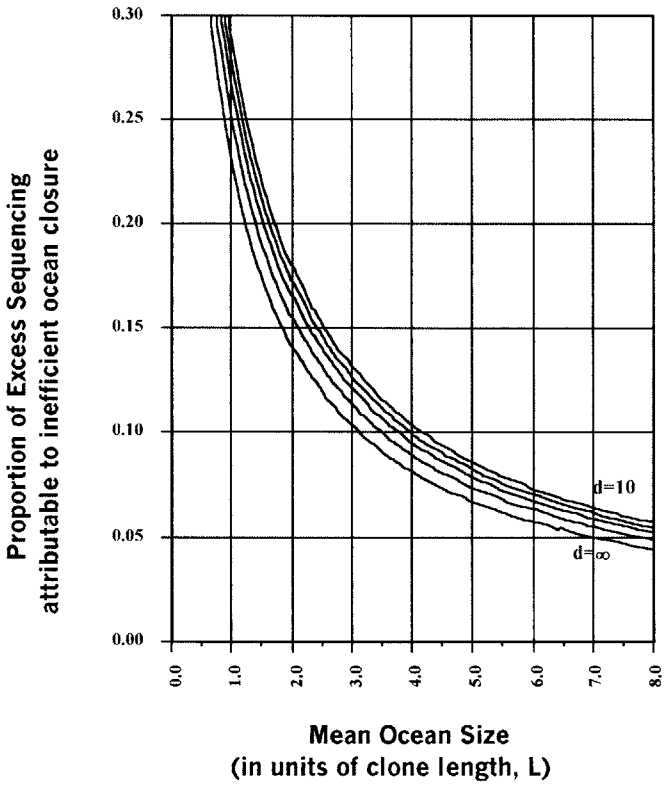

Figure 2.6. Curves show the proportion of excess sequencing attributable to inefficient ocean closure, R*(d, ω). This is obtained by subtracting the proportion of excess sequencing for an optimal tiling path in a library of depth d, which is 1/(d-1). That is, R*(d, ω) = R(d, ω) - (1/d-1).

Figure 2.7 shows the number of unidirectional walking steps required to cover 90% of the genome. The number of unidirectional steps is equal to slightly more than twice the mean ocean size. Since actual walking proceeds bidirectionally, it is thus useful to restate the observation in these terms: *The number of bidirectional walking steps to cover 90% of the genome is approximately equal to the mean ocean length.* (The precise correspondence depends on the library depth d. For example, the number of bidirectional walking steps is roughly 1.1 times the mean ocean length when d = 15.)

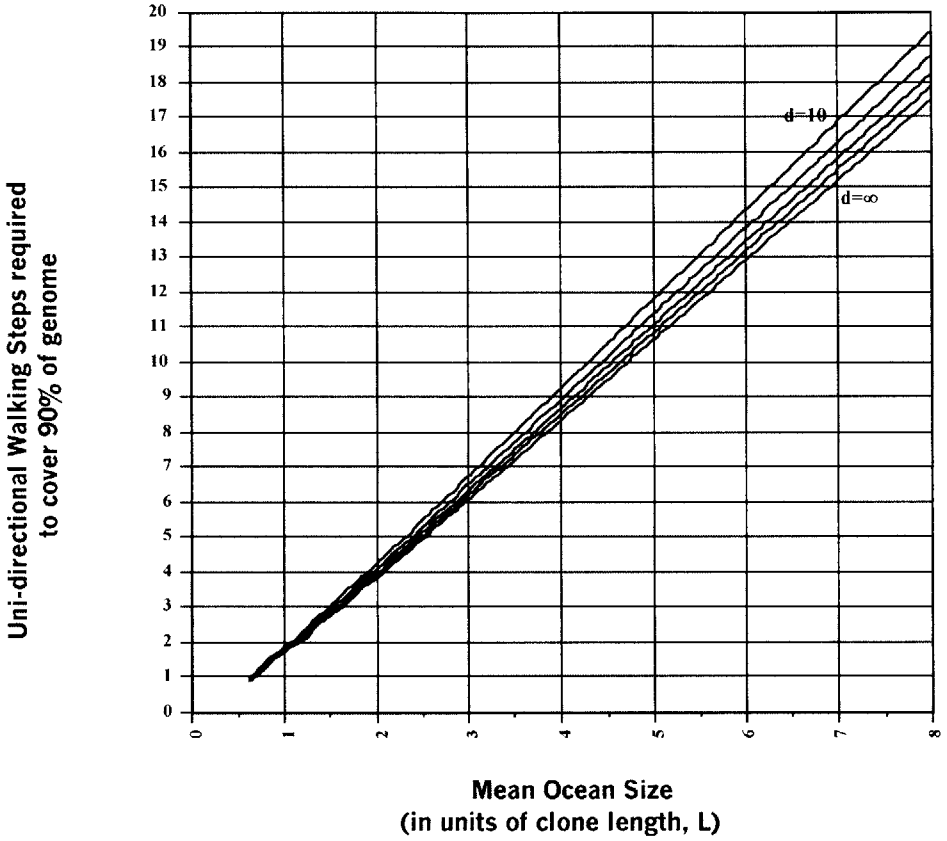

Figure 2.7. Walking Steps. The number of unidirectional walking steps required the cover 90% of the genome, as a function of mean initial ocean size ω for various library depths d = 10, 15, 25, 50 and ∞. In actual practice, walking occurs in both directions. The number of bidirectional walking steps is thus half as many.

If the number of steps shown in the Figure 2.6 is doubled, the proportion of genome sequenced rises to 98-99%. (More precisely, the proportion remaining uncovered is 1% x $(1-\pi)^{-1}$.)

Thus: *The number of bidirectional walking steps to cover 98% of the genome is approximately equal to twice the mean ocean length.*

From Figures 5, 6, and 7, one can readily assess the increase in redundant sequencing entailed in covering the vast majority of the genome in a given number of steps.

4.2 Using Smaller Clones to Close Gaps

The major inefficiency in walking arises from instances in which a clone of length L=1 must be sequenced to close a small ocean; most of the sequencing is redundant. This observation suggests a simple improvement: Use a second BAC library with smaller inserts to close small oceans.

Specifically, suppose that we have two BAC libraries in which the clones have been end-sequenced: our original library B with insert size L = 1 and depth d, as well as a second library B' with insert size L' < 1 and depth d'. We will continue to assume that walking from each seed clone proceeds unidirectionally to the right (but see below concerning this point). At each walking step, we would first search our database to see if there is a clone from B' that spans the remaining ocean (based on its end sequences) and, if none is found, we would select the minimally overlapping clone from B (which either closes the ocean or extends the walk). In this manner, we would aim to select the smallest clone capable of closing the ocean.

We can adapt the mathematical analysis above to calculate the proportion of redundant sequencing. (To simplify the statement of the result, we will consider only the case in which clones from B typically extend the walk farther than clones from B'—that is, $1-(1/d) > L'-(1/d')$. This will be true unless L' is close to 1, in which case the second library adds little anyway.)

Proposition 3. Let B and B' denote BAC libraries as above. Suppose that we initially seed the genome with non-overlapping clones from B to coverage π and then walk as above, using clones from B' to close oceans whenever possible. Let $\omega = (1-\pi)/\pi$. Under AEO and ACO, we have:

(i) Let the random variable J denote the sum of the lengths of the clones used to close an ocean. As before, the expected proportion of redundant sequencing is

$$R(d, \omega) = \frac{(E(J)+1)}{(\omega+1)} - 1$$

(ii) With $p_{d,\omega}$ defined as in Proposition 1, we have

$$E(J) = \frac{1 - p_{d',(\omega/L')}(1 - L')}{p_{d,\omega}}$$

The formula reduces to that in Proposition 1 if the smaller-insert library B' has no clones (d'=0), since $p_{0,\omega/L'} = 0$.

Proof. We proceed as in the proof of Proposition 1. Part (i) again follows by a simple application of Wald's equation. For part (ii), we calculate E(J) by considering a single walking step and applying recursion. With probability $p_{d',(\omega/L')}$, the ocean can be closed with clone from library B', resulting in a total clone length of L'. With probability $p_{d,\omega} - p_{d',(\omega/L')}$, the ocean can be closed with a clone from library B but not B', resulting in a total clone length of 1. With probability $1 - p_{d,\omega}$, the walking step cannot close the ocean and remaining ocean having the same exponential distribution. It thus follows by recursion that:

$$E(J) = (p_{d',(\omega/L')})\, L' + (p_{d,\omega} - p_{d',(\omega/L')})\, 1 + (1 - p_{d,\omega})\,(1 + E(J)).$$

The result follows by solving for E(J).

How much efficiency is gained by using a second library B' with smaller-insert clones? Figure 2.8 shows the proportion of redundant sequencing when an initial library B with depth d = 15 is supplemented with a second library B' with clones whose inserts are half as long ($L' = \frac{1}{2}$) at various depths d'.

The savings are substantial—for example, decreasing the proportion of redundant sequencing from about 23% to 14% for mean ocean length 2 and from about 18% to 12% for mean ocean length 3. The redundant sequencing is considerably closer to the best possible level obtained with a minimal tiling path, roughly 7.1% for a library with d=15 (indicated by the dashed line in Figure 2.8). Indeed, using a library with half-size clones roughly halves the redundant sequencing R' caused by inefficient closure of oceans.

Notably, the second library B' does not need to have high depth to have a major impact. A library with depth d = 5 is only slightly less effective than a library with infinite depth. This makes intuitive sense, since even relatively low depth assures the existence of clones suitable for covering very small oceans.

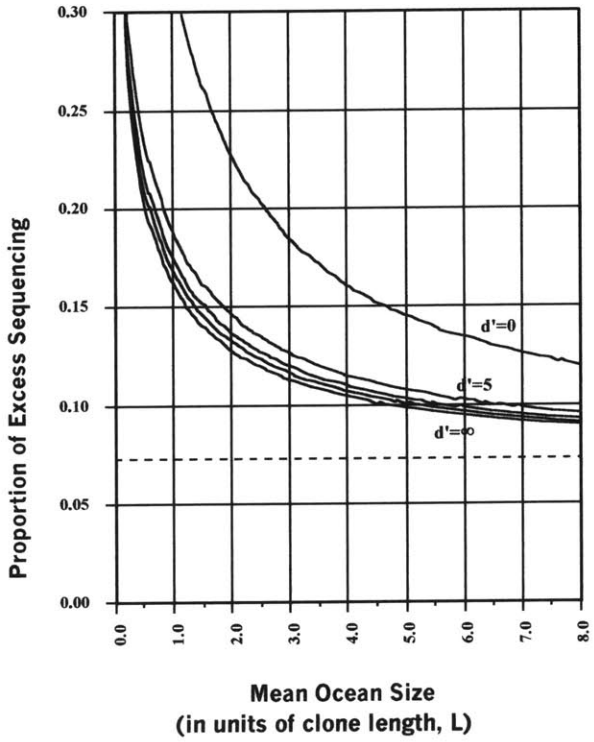

Figure 2.8. Proportion of excess sequencing, for walking with two BAC libraries. The first library B has clones of insert size L = 1 and fixed depth d =15, while the second library B' has clones of insert size L' = $\frac{1}{2}$ and various depths d'. Walking is performed as described in the text. The curves show the proportion of excess sequencing, as a function of mean initial ocean size ω and for various library depths (d = 10, 15, 25, 50 and ∞) for the second library B'.

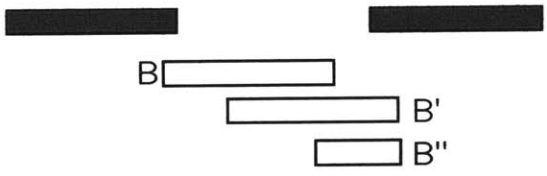

Figure 2.9. Schematic illustration of the situation of an ocean being closed by two large clones (B and B'), when one large and one small clone (B and B'') would suffice.

The results above concern a second library B' with insert size L' = $\frac{1}{2}$. In fact, this choice of insert size L' is nearly optimal for minimizing the amount of redundant sequencing R.

Specifically, one can show by straightforward calculus that the value of L' that minimizes R only changes from 0.55 to 0.50 as the mean ocean size ω goes from 1 to ∞.

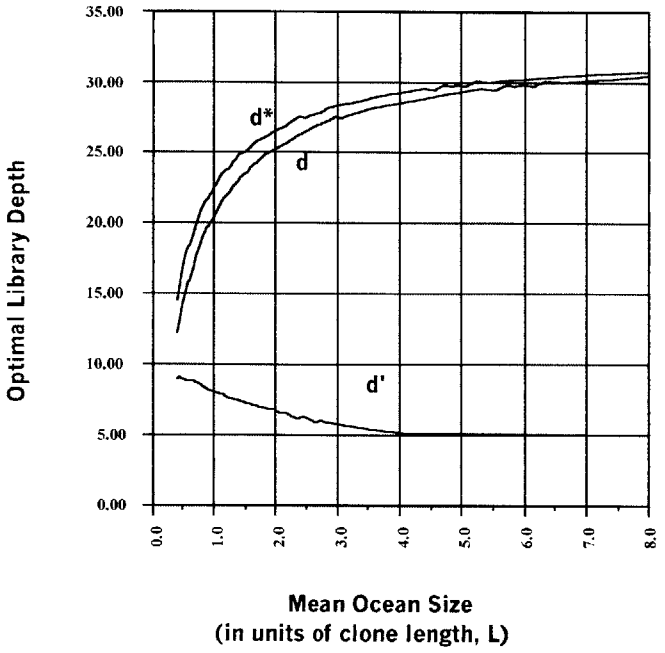

Figure 2.10. Optimal library depth to minimize total cost, assuming that the ratio of the cost of obtaining the complete sequence of a BAC to the cost of sequencing both ends of a BAC is ρ = 1000:1. The top curve (marked d*) shows the optimal depth when using a single library. The two lower curves show the optimal depths d and d' when using two library B and B', as described in the text. The optimal depths depend on the mean initial ocean size ω. The depth d' of the small insert library was constrained to be at least 5, to ensure that the existence of a properly situated, small-insert clone at the end of the vast majority of islands, consistent with the assumption ACO, described in the text).

The analysis above assumes that walking from each seed clone proceeds unidirectionally to the right. The fact that walking is actually bidirectional slightly complicates the situation: If the clones for a given round of walking were all sequenced simultaneously, we would sometimes cover oceans inefficiently by walking with large clones at both ends, when using one large and one small clone would suffice (Figure 2.10). The additional excess sequencing that would result from such occurrences can be shown to be:

$$\rho = (1-L')\left\lfloor \sum_{i=0}^{\infty} p_{d',\omega/L'}(1-p_{d,\omega})^{2i+1} \right\rfloor / (\omega+1) = (1-L')\left\lfloor \frac{(1-p_{d,\omega})}{(2-p_{d,\omega})} \frac{p_{d',\omega/L'}}{p_{d,w}} \right\rfloor / (\omega+1)$$

(Briefly, the first term is the excess contribution to J from sequencing a large clone instead of a short clone, the second term is the proportion of oceans that should properly be closed with one large and one small clone (these are precisely the oceans which would be closed by unidirectional walking with an odd number of large clones followed by a small clone), and the denominator (ω+1) occurs as in the calculation of R in 2(i) above.) For typical values of interest (for example, $d \geq 15$, $d \geq 5$, $L=\frac{1}{2}$, and ω = 1-5), the additional excess sequencing would be in the range of 3%.

It is possible to do much better than this by exploiting the fact that the BAC clones will not all be sequenced simultaneously. As shotgun-sequence is obtained for each large clone B, one can automatically check the sequence to see whether any subsequent large clone B' can be replaced by an ocean-closing small clone B" (as in Figure 2.9). Provided that B' did not pass through shotgun sequencing simultaneously with B, one can thereby avoid the cost of sequencing a larger clone when a smaller clone will suffice. (One may have incurred the cost of preparing a small-insert library from B', but this is small relative to the cost of sequencing.) The additional excess sequencing is therefore only $\alpha\rho$, where $\alpha < 1$ is the proportion of clones in a given walking step that are processed in parallel through shotgun sequencing. The proportion α depends on the workflow of the sequencing operation—specifically, on the amount of work-in-process (WIP) in the shotgun sequencing phase. Since the shotgun sequencing phase is relatively rapid (the elapsed time for picking, growing, preparing and sequencing the small-insert clones is typically on the order of a week), the WIP tends to be relatively small. Even if the proportion of clones simultaneously passing through shotgun sequencing is as high as α = 10%, the additional excess sequencing owing to the occasional failure to use a small clone is only $\alpha\rho$ = 0.3%. In short, the cost is small and does not substantially impact the advantage of using a smaller library. The results derived under the assumption of unidirectional walking are thus not far off.

4.3 Optimizing Clone Library Depth

What is the optimal depth of a BAC library to be used for sequencing a genome by walking? One can decrease the cost of redundant sequencing by using deeper BAC libraries, but one incurs the cost of end-sequencing a larger number of BAC clones. Beyond some point, there are diminishing returns to increasing the depth of the BAC library.

The optimal library depth depends on the relative costs of sequencing entire BAC clones versus sequencing BAC ends. With current laboratory procedures and economics, this ratio is in the neighborhood of ρ = 1000/1. (Roughly 4000 shotgun sequencing reactions are needed to

sequence a 200 kb BAC. Two sequencing reactions are required to sequence its ends, although these reactions are considerably more expensive because sequencing directly from a BAC template requires higher concentrations of reagents.) Given the cost ratio ρ, the optimal library depth is the value d that minimizes the sum of the costs of BAC-end sequencing (proportional to d) and redundant BAC sequence (proportional to $R(d,\omega)$).

Figure 2.10 shows the optimal library depth when using a single BAC library (curve denoted d*). The optimal depth increases with the initial mean ocean length ω, or equivalently decreases with the initial proportion π of the genome covered. The optimal depth d ranges from 22.5 to 30.8 as ω increases from 1 to 8, approaching an asymptotic limit of 32.8 as $\omega \to \infty$. (One can show that the optimal library density approaches $\sqrt{\rho}+1$ as $\omega \to \infty$, by using straightforward calculus to minimize the asymptotic total cost.) The fact that the optimal d is smaller for small ω makes intuitive sense, since a dense library offers more limited advantage when most oceans are fairly small.

Figure 2.10 also shows the optimal library depths d and d' when using a two BAC libraries with respective inserts sizes of 1 and $\frac{1}{2}$. The optimal value of d of the large insert library is only slightly lower than in the previous case, while the optimal value of d' is in the neighborhood of 6-7 for relevant values of ω.

In fact, the optima are relatively broad. Overall, it seems reasonable to use libraries B and B' with respective depths of d = 20 - 25 and d' = 6 – 7.

4.4 Seeding the Genome

We next turn to the issue of generating a collection of non-overlapping seed clones. The analysis above assumes that seed clones are chosen such that initial ocean sizes are exponentially distributed (AEO). How close can one come to this in practice?

The most straightforward experimental approach is to select seeds sequentially from a list of random clones. Each clone is selected and sequenced, subject only to the condition that it does not overlap (as seen from its end sequences) with a previously selected seed clone. In this fashion, one can progressively seed the genome with random, non-overlapping clones.

This stochastic process is known in the literature as the 'parking process' because it is equivalent to cars of constant size (clones) sequentially choosing random parking spots along a long street (genome). Each car is permitted to park in its chosen parking spot provided that a previously parked car does not block it; otherwise, it must drive away.

The resulting 'parking distribution' is relevant to many physical processes and has been well studied (Krapivsky, 1992). Suppose that one has processed a list of random clones comprising a sub-library covering the genome to depth t. The expected proportion of genome covered with seed clones will be $\pi(t)$ and the distribution of oceans sizes will be $\omega(x,t)$, where:

$$\pi(t) = \int_0^t F(\tau)d\tau \ ,$$

$$\Omega(x,t) = \begin{cases} [t^2 F(t)\exp(-(x-1)t)]/\pi(t), & x > 1 \\ [2\int_0^t \tau F(\tau)\exp(-x\tau)d\tau]/\pi(t), & x \le 1 \end{cases}$$

$$F(t) = \exp[-2\int_0^t \frac{1-\exp(-\tau)}{\tau}d\tau]$$

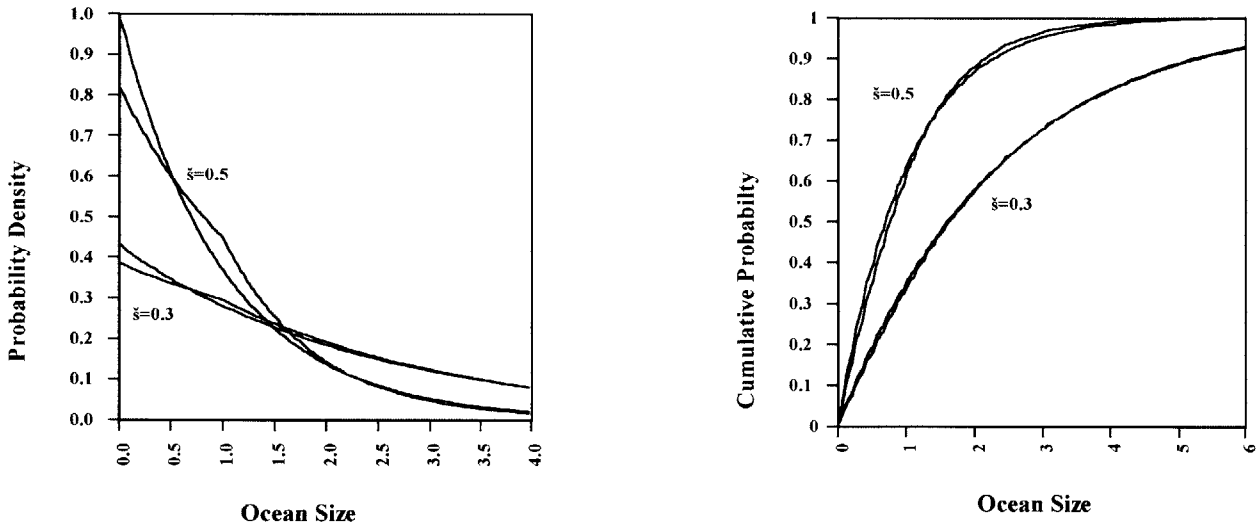

Figure 11. Comparison of Parking and Exponential Distributions. (A) Probability density and (B) Cumulative probability distribution, for $\pi = 0.3$ and $\pi = 0.5$. The kinked curves are the parking distribution; the smooth curves are the exponential distribution with the same mean.

The parking distribution of ocean sizes is very close to the exponential distribution for low coverage $\pi(t)$ and is still reasonably close for coverage $\pi(t) = 50\%$ (Figures 11 a,b).

The expected proportion of excess sequencing under the parking distribution can be calculated as in Proposition 2. The difference Δ between the excess sequencing expected under the exponential distribution and the parking distributions is shown in Figure 2.12. The parking distribution entails slightly less excess sequencing than the exponential distribution (i.e., $\Delta > 0$), because it has fewer extremely small intervals (as see from Figure 2.10a). The difference Δ is quite small over the range of interest, being about 0.6% for $\omega = 1$ and decreasing as ω increases.

It should be noted that the parking process cannot fully cover the genome: As the coverage increases, the remaining spaces become smaller and the proportion of cars turned away increases. When the remaining spaces are all smaller than a car length, no more cars can park. This happens at the so-called 'jamming limit', which occurs at $\pi = 0.747597....$ Sequential selection of strictly non-overlapping clones thus cannot provide seed coverage beyond this point.

Fortunately, we are not interested in seeding to more than 50% because the cost of excess sequencing begins to sky-rocket for $\pi > 50\%$. Seeding the genome to 50% requires starting with a random list of clones covering the genome to depth 1.15 (because $\pi(1.15) \sim 0.5$).

We briefly mention some other approaches to seeding the genome, but do not analyze them in detail.

(i) Simultaneous seeding. One could start with a collection of random clones, prepare small-insert shotgun library from each, and perform a small amount of sequencing from each shotgun library (for example, covering the clone to an average depth of 0.5-fold). Such 'sample sequencing' should be sufficient to detect any significant overlap between clones. One could then select a maximal set of non-overlapping clones. If the collection of random clones covers the genome to depth t, a maximal non-overlapping set will cover a proportion $\pi \sim t/(t+1)$ of the genome and the oceans will be very nearly exponentially distributed with mean size 1/t. The approach allows the genome to be seeded to higher initial coverage, but the sample sequencing is more expensive than end-sequencing and it must all be performed in advance. Still, there may be situations in which such an approach may be desirable.

(ii) Contig-based seeding. One could initially fingerprint the entire BAC library to construct non-overlapping contigs, which could be used as the seeds for subsequent walking. Assuming that the

contigs were separated by roughly exponentially distributed oceans, the situation could be analyzed in the same manner as above.

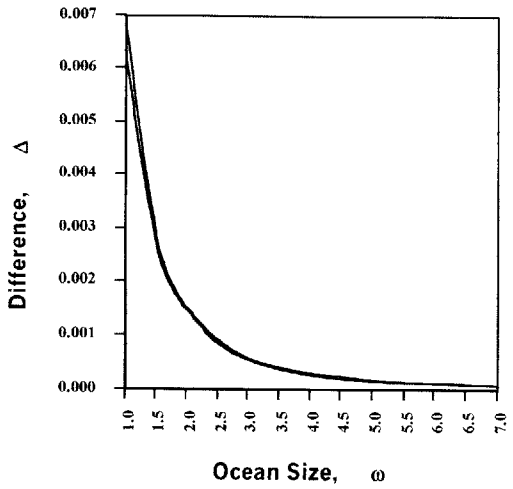

Figure 2.12. Difference between exponential and parking distributions. Graph shows difference Δ between R_{exp}, the proportion of excess sequencing under exponential distribution, and $R_{parking}$, the proportion of excess sequencing under parking distribution. That is, $\Delta = R_{exp} - R_{parking}$. Two curves are shown, corresponding to d = 15 and d =50. The difference Δ is seen to be largely insensitive to d.

5. Simulations

The simple formulas above are premised on two simplifying assumptions (AEO and ACO). We tested whether the formulas provided reasonable approximations by performing extensive simulations.

The simulations were performed as follows. Libraries were generated by randomly selecting starting points for BAC clones (L = 2 x 10^5 bp, L' = 1 x 10^5 bp) from a mammalian genome (3x10^9 bp) using a uniform distribution to achieve the desired depths d and d'. An initial selection of seed clones was generated as in the parking problem, by considering the clones in a random order and accepting successive non-overlapping clones until the desired proportion π of the genome was reached. (We confirmed that the simulation yielded oceans sizes closely fit the parking distribution (not shown).) Walking was then performed by selecting minimally overlapping BACs as described above. For each choice of parameters, a total of 40 simulations

were performed and the results were averaged. Simulations were performed for parameters throughout the range of interest, at the parameters d $\in$ {10, 15, 25, 50, ∞} and ω = {1.0, 1.5, 2.0, 2.5, . . . , 7.0}. We calculated $R_{sim}(d,\omega)$, the average proportion of redundant sequencing over the simulations.

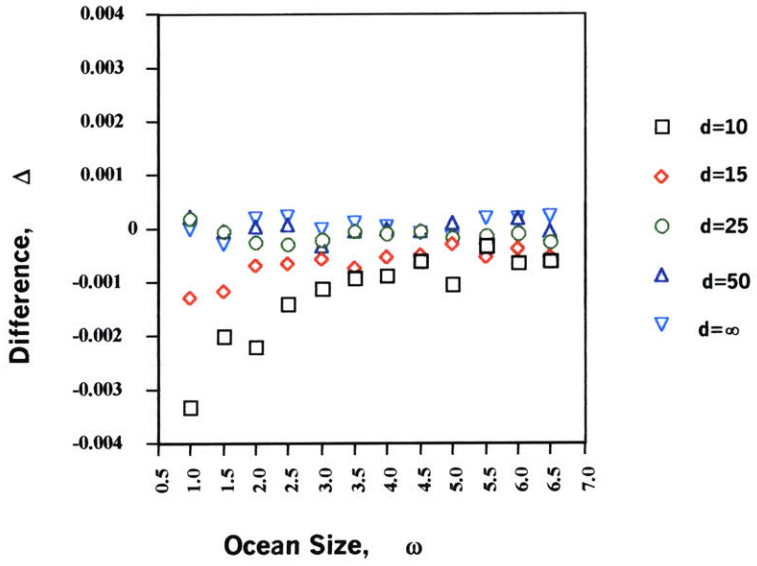

Figure 2.13. Difference between formulas and simulations. Plots show the differences Δ between the proportion of excess sequencing predicted under the parking distribution ($R_{parking}$) and the proportion of excess sequencing observed in (A) simulations in which the seed clones and walking clones are chosen from a randomly distributed library (R_{sim}). Results are shown for various values of d and ω.

We also performed simulations mimicking the ACO assumption, in which the seed clones were selected as above but walking steps then occurred under the assumption that the minimally overlapping clone overlapped by exactly 1/d. We calculated $R_{sim,ACO}(d,\omega)$, the average proportion of redundant sequencing over these simulations.

We compared the simulation results to $R_{parking}(d,\omega)$, the excess sequence under the parking distribution predicted by the formula in Proposition 2. The difference $\Delta(d,\omega) = R_{parking}(d,\omega) - R_{sim}(d,\omega)$ is shown in Figure 2.13. The difference is negligible (on the order of the standard error of the mean from the simulations), except for small d and small ω. The discrepancy for small d and ω is readily seen to be due to ACO, by examining the difference $\Delta'(d,w) = R_{parking}(d,\omega) - R_{sim,ACO}(d,\omega)$ shown in Figure 2.14. This difference is negligible over the entire range.

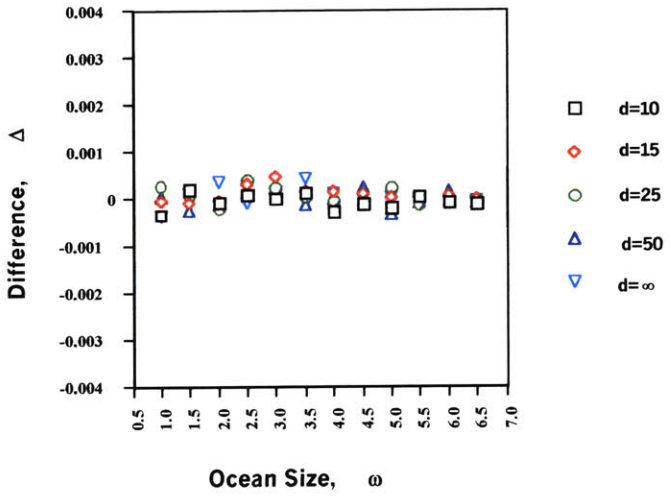

Figure 2.14. Difference between formulas and simulations. Plots show the differences Δ between the proportion of excess sequencing predicted under the parking distribution ($R_{parking}$) and the proportion of excess sequencing observed in simulations in which seed clones are chosen from a randomly distributed library, but walking steps are then made under the ACO assumption ($R_{sim,ACO}$). Results are shown for various values of d and ω.

We also studied the proportion of genome covered after k steps (k = 1, 2, . . ., 10), comparing the prediction under the exponential distribution (Proposition 1(iii)) to the results seen in the simulations above. In the range of interest, the predicted proportion of genome covered at each stage was within 0.5% of the results from the simulations (not shown). Finally, we simulated the situation of two libraries, where d and ω were as above and the smaller library has depth d' = 5. Again, the difference ω was extremely small (not shown).

In summary, the simple formulas derived under the assumption of AEO and ACO are sufficiently close for practical purposes. The differences are quite small and have no impact on the key conclusions.

6. Conclusion

Sequencing a genome by walking is an inherently serial process. One seeds the genome to an initial coverage π and then engages in sequential rounds of walking.

Completing the task in a reasonable amount of time requires a high degree of parallelism—that is, a high density of seed clones from which walks proceed outward in both directions. The number of such walking steps to cover 90% of the genome is roughly equal to the initial mean ocean size ($\omega = (1-\pi)/\pi$), while the number of steps to cover 98% is roughly twice as large.

There is a clear tradeoff between the number of walking steps and the cost of redundant sequencing. As the mean ocean size ω grows from 1 to 2 to 3, the proportion of redundant sequencing decreases from 32% to 23% to 19%. (These values correspond to a library with depth d=15; they are about two percentage points lower for libraries with d=25.)

One solution is to decrease the cycle time required for each walking step, making it feasible to take more serial walking steps within the desired time frame. Since the time required to prepare and validate high-quality shotgun libraries represents a significant component of the cycle time, one might develop efficient and inexpensive methods to prepare shotgun libraries from all clones in the BAC library in advance of sequencing. (The storage requirements are modest, since each shotgun library is stored as a ligation mixture in a single test tube until required.) One could then rapidly initiate each consecutive walking step. Implementing such an approach would require significant streamlining or automation of existing procedures for preparation of shotgun libraries, but may be feasible.

A complementary solution is to employ a second BAC library with smaller clones, with roughly half the insert size. This approach dramatically improves efficiency. For the cases involving a 15-fold library above, the proportion of redundant sequencing is 18%, 14% and 12%. This is much closer to the best possible result obtainable with a 15-fold library of roughly 7.1% (=1/14). Importantly, the vast majority of the efficiency is obtained using a second BAC library having relatively modest depth — for example, d' = 5.

We should note that our analysis ignores certain biological and experimental issues.

(i). Variable insert size. We have assumed that the BAC libraries have inserts of constant size. A BAC library with variable insert size may be more efficient than one with constant size, because one has the potential to optimize the size of the BACs used to close oceans. Indeed, our strategy of using of two BAC libraries with constant inserts of size L and L' is formally equivalent to using a single BAC library with variable insert size equal to either L or L'. In principle, BAC libraries with variable insert size can be extremely efficient: a BAC library with highly variable insert size and infinite depth would allow one to close gaps with essentially no wasted sequencing.

In practice, however, current BAC libraries have a fairly tight size distribution and thus the effect is rather modest. The human RPCI-11 library, for example, has insert size that is roughly normally distributed with mean 160 kb and coefficient of variation (CV, defined as ratio of standard deviation to the mean) of somewhat less than 10%.

We performed simulations to compare the proportion of redundant sequencing in two situations cases: (i) a single library having mean insert size 1 with the insert size either being constant or normally distributed with CV=10%, and (ii) two libraries having mean insert sizes 1 and 0.5 with the insert sizes either being constant or normally distributed with CV=10%. As expected, the variable libraries were slightly more efficient than the constant library. For $\omega = 1$, the absolute difference was roughly 4% in the case of a single library (d=15) and 2% in the case of a two libraries (d=15, d'=5). For larger ocean sizes ω, the absolute differences are even smaller. In short, the effects are thus fairly small and the key conclusions concerning the value of a library containing smaller clones remain valid.

An interesting open question is to find the optimal distribution of insert sizes for a BAC library of a given depth.

(ii) Cloning bias. We have assumed that there is no cloning bias in the genomic library. Cloning bias is known to occur in many cloning systems, but the nature and distribution of cloning bias is poorly understood and therefore difficult to model. Severe cloning bias against some regions would clearly lead to larger initial oceans, as well as lower effective library depth, in these regions. One could conceivably model the genome as composed of large blocks with different cloning bias.

(iii) Missing end sequences. We have assumed that each BAC has been sequenced at both ends. Current BAC-end sequencing projects sometimes fail to generate one of the end-sequences owing to technical failures. Such 'single-ended' BACs are less inefficient because one might select two such clones to walk from each side of an ocean and be unaware (because of the lack of the opposite sequence) that each clone suffices to close the ocean on its own. The impact of the problem can be assessed through either mathematical analysis or simulation. The problem itself can be overcome simply by sequencing additional BAC clones to achieve the desired depth d in 'double-ended' clones.

(iv) Repeat sequences at BAC ends. We have assumed that each BAC has unique sequence at both ends to permit unambiguous recognition of overlap. However, a clone end may consist

entirely of a repeat sequence, which cannot be used for walking. If the repeats are relatively small and fairly uniformly distributed across the genome, the problem can be overcome simply by end-sequencing a larger number of BACs to obtain enough clones with unique sequence at both ends (as in the case of 'single-ended' BACs due to technical failure). If repeats are unevenly distributed across the genome, the problem will lead to under-representation of clones in the repeat-rich regions (as in the case of cloning bias). If very long repeats occur, they will prevent the initiation of walks from within the repeat and may necessitate larger overlap. In general, the most practical solution is to increase the depth of the library used for end-sequencing. (See also, Siegel et al. 1998).

We have focused on developing simple models to provide insight into the problem of sequencing a genome by walking. In actual application, one will confront complications such as those above. The best solution is to perform specialized simulations. Our results however, help define the key issues and tradeoffs and should be helpful in conceptualizing how to design a clone-based genomic sequencing project.

Chapter 3

WHOLE GENOME SHOTGUN ASSEMBLY

1. Introduction

In the previous two chapters we talked about two important variations in the clone-by-clone approach to sequencing a genome, namely physical mapping, and clone-end walking. The topic of this chapter is the alternative approach to clone-by-clone sequencing — whole genome shotgun sequencing. We describe an approach for assembling in the computer the data produced by shotgun sequencing; we demonstrate the performance of this approach by developing and testing *ARACHNE*, a prototype system for shotgun assembly.

1.1 Previous Work

Previous work on whole-genome shotgun assembly is limited. *PHRAP* (http://www.PHRAP.com) is virtually the only program available, and it is mainly targeted to the assembly of relatively short genome segments such as a BAC clone (roughly 200,000 nucleotides long). The algorithm uses a greedy scheme for connecting reads into contigs based on the quality of alignment between two reads, and on the relative likelihood that the two reads truly overlaps, as opposed to lying in a repeat with 95% fidelity between the two copies. These likelihoods are calculated using the quality scores associated with each read position whereby low quality score positions are more likely to be wrong, and therefore reads that truly overlap are more likely to differ in the positions where at least one of them has low quality bases. On the other hand reads that do not truly overlap, but instead come from two copies of the same repeat, may differ in positions of high, as well as low, quality. *PHRAP* does not make use of forward/reverse linking information between pairs of reads that come from two ends of the same insert (*earmuff* links, as we defined them in the Background section). It is generally considered to solve with great quality

the problem of assembling BAC clones, but it gets too slow, and inappropriate to use, for genomes or genomic segments of length more than a few megabases.

Anson and Myers (1999) describe an algorithm for whole genome assembly, focusing on creating the assembly of a region of DNA between two reads that are linked with an earmuff. Their algorithms strive to minimize the number of queries to the database of reads, where each query asks for the set of reads that overlap with a particular read. They test their algorithm on a simulated model of the genome.

The most important breakthrough to date in shotgun assembly is the recent successful whole genome assembly of Drosophila melanogaster (Myers et al. 2000). This assembly was achieved by a software system that shares ideas with the system described in this chapter, and was developed independently. We note that the assembly of Drosophila took around one week on an 8-node ES40 platform (Myers et al. 2000), which was due to an $O(n^2)$ number of pairwise read alignments, a step that our system avoids by a hashing mechanism that we describe below.

1.2 Problem Description

The general problem can be stated as follows: the goal is to reconstruct an unknown source sequence (the *genome*) on {A, C, G, T} given many random short segments from the sequence, the *shotgun reads*.

A *read* is a subsequence of nucleotides of length around 500, taken from a random place in the genome. The *orientation* of the read is either *forward*, in which case the read is taken from the forward strand of the genome, or *reverse complement*, in which case the read is taken from the reverse complement strand. Recall that the reverse complement strand is essentially the same as the forward strand, except each nucleotide is substituted with its Watson-Crick complement: A $\leftrightarrow$ T, and C $\leftrightarrow$ G.[1] The direction of a read is not known — reads are *unoriented*.

[1] We call that strand the *reverse* complement because chemically it runs 5' $\rightarrow$ 3' instead of 3' $\rightarrow$ 5'. Also it is important to note that genes (and presumably other elements) lie either in the forward, or reverse complement direction. That is, RNA polymerase works only 3' $\rightarrow$ 5' and therefore grows a pre-mRNA strand during transcription, in the 5' $\rightarrow$ 3' direction. The pre-mRNA in turn, is always "interpreted" in the direction 5' $\rightarrow$ 3' during splicing and translation.

The read is not an exact subsequence of the genome. It contains errors of two kinds: *base substitutions*, and *indels*. Base substitutions occur with a certain moderate average frequency, typically 0.5-2%.[1] The number of them on each read is Poisson distributed and the occurrences are uniformly distributed along the read.[2] Indels are either insertions, or deletions of nucleotides. They occur with the same distribution, but roughly an order of magnitude less frequently, than base substitutions.

Some reads come with additional linking information. Recall the description of an *earmuff* link in the Background section. A pair of reads can be taken from the two ends of the same size-selected insert. Therefore the relative orientations, and approximate distances between pairs of reads, are known. Depending on the vector system and size-selection, we can have read pairs coming from short plasmid inserts, of length 2,000-3,000 up to 10,000-12,000. Or they can come from cosmids, of length around 40,000. Finally they can come from BACs, which are around 150kb *long*. A combination of all these kinds of linked reads is probably more powerful than any one kind of links by itself.

2. Algorithms

ARACHNE consists of two parts: (i) the creation of a graph G of overlaps between pairs of reads of the shotgun data, and (ii) processing of G for the purpose of constructing supercontigs of mapped reads.

2.1 Creation of Overlap Graph

The input to *ARACHNE* is a list of reads taken from the genome. We will denote this list by $R = (r_1,\ldots, r_N)$ where N is the number of reads. Each read r_i has an associated length l_i that is typically < 1000. It may also have a link to another read r_j at specified distance d_{ij} and known relative orientation, in case both reads were taken from the endpoints of the same clone (*earmuff* link). Finally, each position of a read may have an associated confidence score (*PHRED* score (Ewing

[1] A read is typically longer than 500, but the nucleotides beyond around 500 are of poor quality. These nucleotides have a higher rate of base substitutions and indels. The associate *PHRED* scores indicate the quality of each nucleotide.

[2] But note again that typically *PHRED* scores give some information on the likelihood of error, for each specific read position.

and Green, 1998) that characterizes the probability that the nucleotide is correct. For the moment we will ignore this additional information and simply view the input as a list R of reads.

The first goal of *ARACHNE* is to create a graph G of overlaps (edges) between pairs of reads (nodes). In order to create G, pairs of reads in R need to be aligned to detect overlaps. However R can be very long, making N^2 alignments infeasible to perform. For that reason we first create a table of occurrences of k-mers (k *long* strings) in the reads, and count the number of k-mer matches for each pair of reads. Subsequently, we perform pairwise alignments between pairs of reads that contain more than a cutoff number of common k-mers.

2.1.1 Table of k-mer Occurrences

The goal of this algorithm is to find the number of k-mer matches in the forward or reverse complement direction, between each pair of reads in R. One way to do that is to (1) obtain all triplets (r, t, v) where r is a read in R, t is the index of a k-mer occurring in r, and v is the direction of occurrence (forward or reverse complement); (2) sort the set of pairs according to k-mer indices t; (3) using the sorted list create a table T of quadruplets (r_i, r_j, f, v) where r_i and r_j are reads that contain at least one common k-mer, v is a direction, and f is the number of k-mers in common between r_i and r_j in direction v.

Reads		**3-mers**
r_1 = ACTAG	→	{(ACT,0), (CTA,0), (TAG,0), (CTA,1), (TAG, 1), (AGT, 1)}
r_2 = AGCTA	→	{(AGC,0), (GCT,0), (CTA,0), (TAG,1), (AGC,1), (GCT,1)}

Sorted Table

Sorted 3-mers	Sorted Directions	Read ID	Read ID
ACT	0	1	
AGC	0	2	
AGC	1	2	
AGT	1	1	
CTA	0	1	2
CTA	1	1	
GCT	0	2	
GCT	1	2	
TAG	0	1	
TAG	1	1	2

Table 1. Example of a sorted hash table of k-mer occurrences in reads.

If a k-mer occurs "too often" it is likely to be part of a repeat sequence and we should not use it for detecting overlap. ("Too often" depends on the depth d. The number of times that a k-mer occurs should be roughly Poisson distributed with mean d. We can use this observation to set a threshold to "exclude" those n-mers that occur "too often.)

This is implemented as follows:

1. The k-mer occurrences (r, t, v) are found and sorted initially into 64 files according to the first three nucleotides of each k-mer (the first six bits of t).

2. Each of the 64 files is short enough to be loaded in memory as a table of k-mer occurrences (r, t, v).[1] A radix sort is performed according to t, and the sorted occurrences are saved in a file. The original file of unsorted occurrences is erased.

3. The 64 sorted files are loaded in memory sequentially, and the table T is incrementally created.

2.1.2 Pairwise Read Alignments

[1] More than 64 files could be used if less memory is available.

Pairwise alignments are performed between reads that contain more than a cutoff number of common k-mers. If k-mers that are more common than a certain cutoff are excluded from being used in the table T, as explained above, then it is guaranteed that only an O(N) number of pairwise alignments will be performed.

Only a small number of base substitutions and insertions/deletions is allowed in an overlapping region of two aligned reads, reflecting the low expected error rates of sequencing for reads. This allows for an efficient alignment algorithm, where a shift is initially guessed between a pair of reads, and subsequently a dynamic programming alignment is performed, that disallows deviations of more than a few characters, from the guessed shift. The output of the alignment algorithm is a quadruplet of positions (b_1, e_1, b_2, e_2) of beginning b_1, b_2, and end e_1, e_2, positions of the detected overlap region, for reads r_i, r_j. If a significant overlap region is detected, (r_i, r_j, v, b_1, e_1, b_2, e_2) becomes a link in the overlap graph G.

2.2 Processing of Overlap Graph and Creation of Supercontigs

The remaining algorithm strive to map the reads as accurately as possible, so that eventually the genome sequence can be retrieved, or at least the mapped reads can be passed to a local assembly program that can routinely retrieve the sequence (for instance *PHRAP*, http://www.PHRAP.com). The information available at this stage is the graph G, and the forward/reverse linking of pairs of reads that come from two ends of the same insert clone.

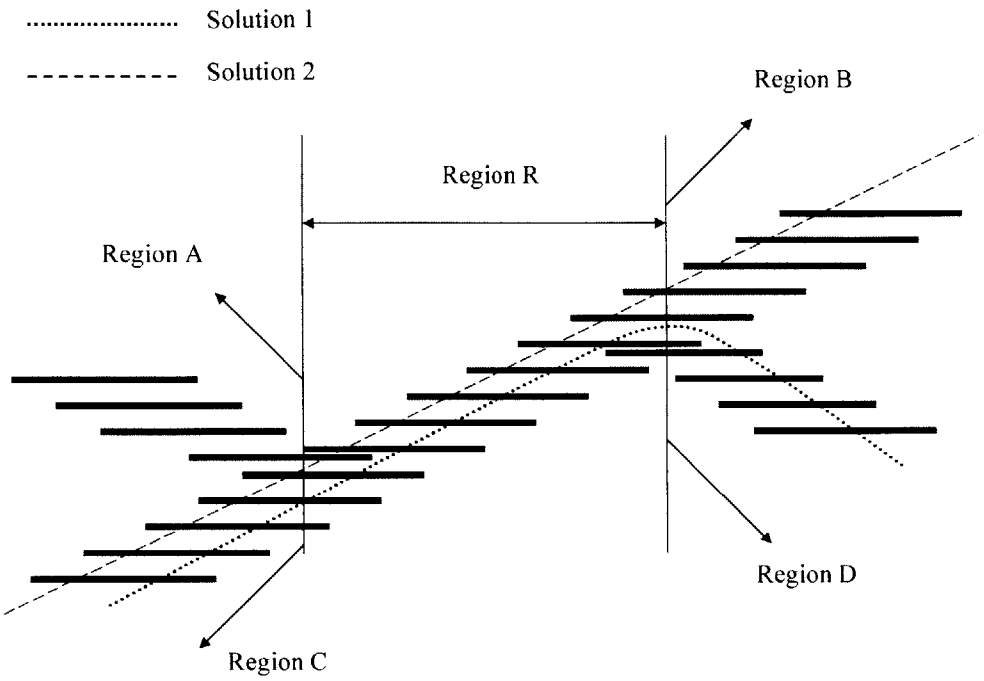

Figure 3.1. Ambiguity created by the presence of repeats. What is the correct assembly of reads? One possibility (solution 1) is that regions C and D flank one copy of repetitive region R, while regions A, B flank another copy. Another possibility (solution 2) is that regions C and B flank one copy, and regions A and D flank the other copy of R.

In the absence of sequencing errors and repeats it would be a simple task to retrieve all retrievable pairwise distances of reads: it is easy to see that G would then be an interval graph.[1] In the presence of repeats, a link between two reads in G does not necessarily imply true overlap. A "repeat link" thus is a link in G between two reads that come from different regions in the genome, and overlap in a repeated segment. The ambiguity created by repeats can be seen in the depiction of paths of overlapping reads in Figure 3.1. Region R is repetitive, flanked by regions A and B on the "left", and C, D on the "right. It is not possible a priori to know whether the path of

[1] An undirected graph $G = (V = \{1, \ldots, n), E)$ is an *interval graph* if there is intervals $(a_1, b_1), \ldots, (a_n, b_n)$ on the real line such that $(i,j) \in E$ if and only if (a_i, b_i) intersects (a_j, b_j). It can be checked in linear time whether a graph is an interval graph (Fishburn, 1985).

reads corresponding to region A, and "entering" into region R, needs to "exit" towards region B or region C.

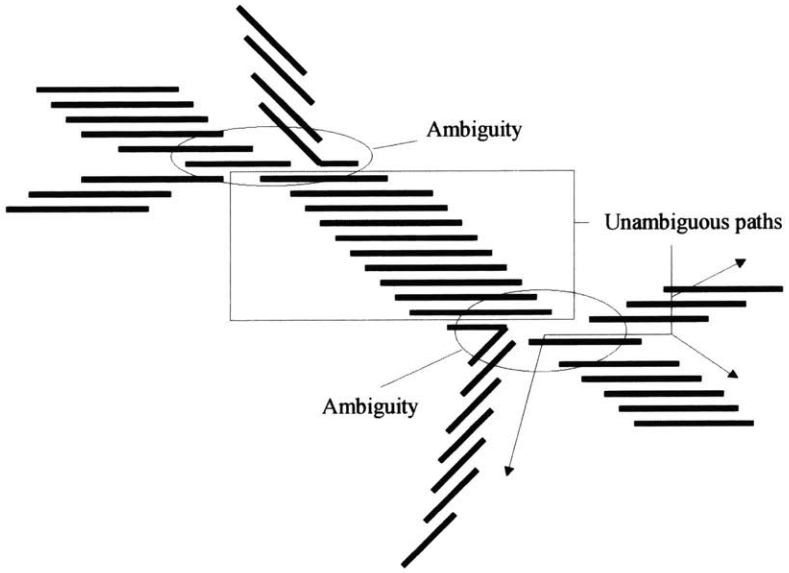

Figure 3.2. A repeat with three copies translates into one contig, with borders to three other contigs on the left, and three on the right.

Some of the repetition in the genome is effectively masked before the creation of G, by throwing away k-mers of high frequency when building the table T. In the algorithms that follow some additional repetitive links are detected and deleted using heuristic rules.

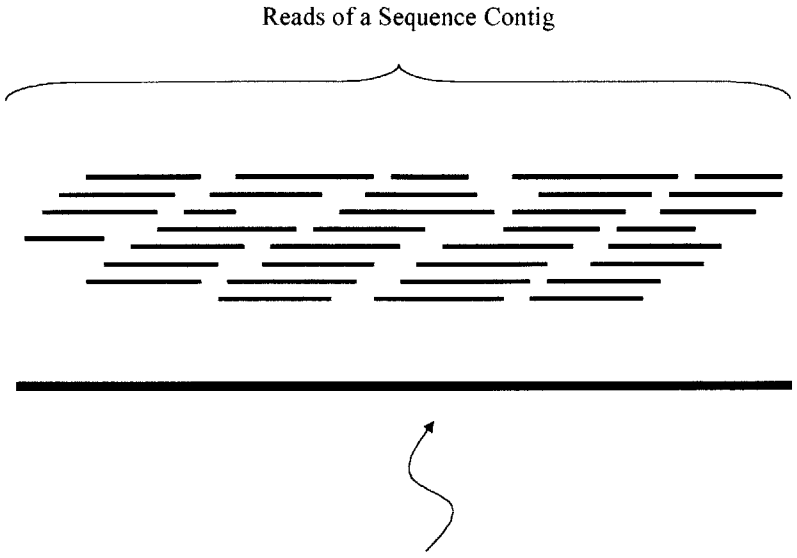

Figure 3.3. A sequence contig is a structure holding an oriented set of reads, each holding a specific position on the sequence contig. It is represented as an interval RED/BLACK tree.

Sequence contigs are formed by merging together pairs of reads that can be merged without ambiguity. This is illustrated in Figure 3.2, where we show a situation created by a repeat that is copied three times. Figure 3.3 shows how a sequence contig looks like. The contigs are finally linked using the forward/reverse linking information between pairs of reads. We should note that in practice the situation is much less neat than the one depicted in Figure 3.2. Repeats are not preserved 100% between copies, and therefore they are not guaranteed to form unambiguous contigs. Certain parts of them that are preserved much less than the cutoff for considering two reads overlapping may form independent contigs. What is worse, in the situation of reads covering repeats (or parts of repeats) that are on that borderline of fidelity (around 95%) between copies, reads from two different copies may be considered overlapping or nonoverlapping depending on the sequencing errors.

2.2.1 Definition of Read "Shifts"

G consists of links of the form $(r_i, r_j, v, b_1, e_1, b_2, e_2)$ where r_i, r_j are reads, $v \in \{0, 1\}$ denoting *forward* and *reverse complement* respectively, and (b_1, e_1), (b_2, e_2) are the intervals of alignment of r_i, and r_j, respectively. Abusing notation sometimes we will omit b_1, e_1, b_2, e_2 and/or v when referring to a link. These links are directed in the sense that $(r_1, r_2) \in G$ does not imply $(r_2, r_1) \in G$.

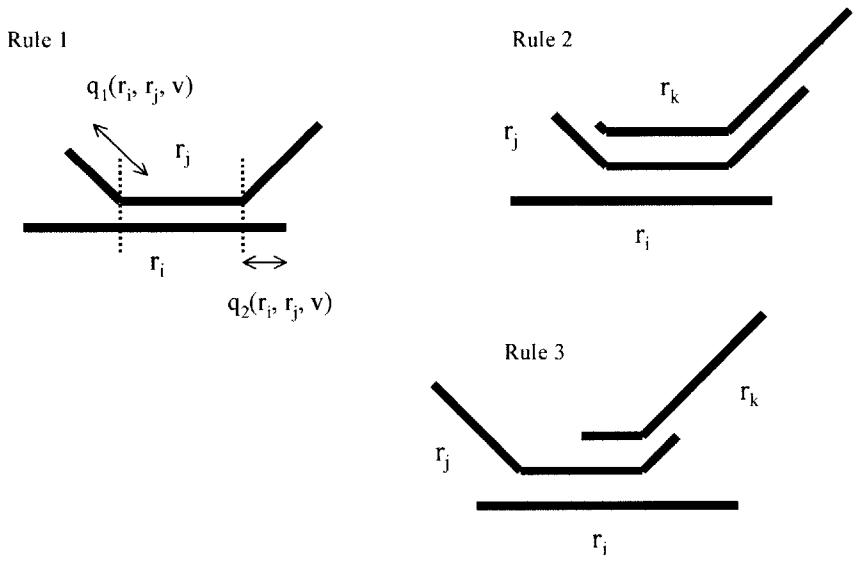

Figure 3.4. Rules for removing repetitive links.

Notice that in the absence of repeats and sequencing errors, $|b_1-e_1| = |b_2-e_2|$. Moreover, in that case we can distinguish cases of the possible relationship between r_i, and r_j:

1. r_j is a subsequence of r_i, and $b_2 = 1$, $e_2 = l_j$.

2. r_j extends r_i to the "right", whereupon $e_1 = l_i$ and $b_2 = 1$.

3. r_j extends r_i to the "left", whereupon $b_1 = 0$ and $e_2 = l_j$.

4. r_i is a subsequence of r_j, and $b_1 = 1$, $e_1 = l_j$.

In case 1 r_i is called a superread of r_j and r_j a subread of r_i. Similarly for case 4.

In the presence of sequencing errors, the above conditions are no longer guaranteed to hold for a pair of reads that truly overlap in the genome. In case 1 above for instance, a few base substitutions and/or insertions/deletions in r_i or r_j near the beginning of r_j can cause b_2 to be > 0. In any overlapping pair of reads, where one of the above three conditions (and the corresponding ones for $v = 1$) does not hold, we say that we detect a *nonaligning end.* The amounts of deviation from the values specified in conditions 1, 2, or 3 for the values b_1, e_1, b_2, e_2 are called the *nonalign values* $q_1(r_i, r_j, v)$, $q_2(r_i, r_j, v)$. In condition 1, $q_1(r_i, r_j, v) = 0$, $q_2(r_i, r_j, v) = b_2 + e_2 - 1$. In condition 2, $q_1(r_i, r_j, v) = l_1 - e_1$, $q_2(r_i, r_j, v) = b_2 - 1$. In condition 2, $q_1(r_i, r_j, v) = b_1$, $q_2(r_i, r_j, v) = l_2 - e_2$. We define $Q(r_i, r_j, v) = \max(q_1, q_2)$. The values q, Q will be important heuristically for the algorithm to decide whether a particular link should be "trusted" as a true link, or taken as a false link resulting from a repetitive segment in the genome. A heuristic constant C_{rep} will usually determine whether a link should be considered true (if $Q < C_{rep}$) or false ($Q \geq C_{rep}$). This applies to definitions of subreads and superreads.

Shifts. In case 2 above, we define the *shift to the right* $s_r(r_i\, r_j) = (l_2 - b_2) - (l_1 - b_1)$, of r_i by r_j. In case 3 above, we define the *shift to the left* $s_l(r_i\, r_j) = b_2 - b_1$, of r_i by r_j.

2.2.2 Discarding Repetitive Links

At this stage certain links of the graph are discarded by the application of heuristic rules. The idea is to remove from G some repetitive links, while at the same time preserving enough of them so as not to "hide" the boundaries between unique sequence and long repetitive regions. Thus mostly short repeats are targeted.

The following rules are used for discarding links from G:

1. *Rule 1.* If $\min(q_1(r_i, r_j, v), q_2(r_i, r_j, v)) \geq C_{rep}$, remove link (r_i, r_j, v).

2. *Rule 2.* If link $(r_i, r_j, v=0)$ is marked, remove any link $(r_i, r_k, v=0)$ where $Q(r_j, r_k, v=0) < C_{rep}$.

3. *Rule 3.* If link (r_j, r_i, v=0) is marked, remove any link (r_i, r_j, v=0) where $Q(r_j, r_k, v=0) < C_{rep}$ and $Q(r_i, r_j, v=0) \geq C_{rep}$.

Rules 2 and 3 above are applied recursively until no more applications are possible.

2.2.3 Sparse Representation of Overlaps and Creation of Contigs

The first main step of the assembly algorithm is the detection of subgraphs of G that have the interval graph property. Such subgraphs are then assembled into sequence contigs. As an initial step, all links (r_i, r_j, v) of G such that $Q(r_i, r_j, v) \geq C_{rep}$ will be marked and not considered when constructing the interval subgraphs. Also, any link (r_i, r_j, v) that involves an overlap between the reads r_i, r_j, of less than a heuristic constant C_{ovl} will be marked.[1] This ensures that all links used will have a statistically significant associated overlap. Finally, any link involving a subread is marked.

Next a sparse representation of the subgraph of unmarked links of G, is created. Specifically, a graph H is created as follows: given an unmarked link (r_i, r_j, $v = 0$) of G, (r_i, r_j, $v = 0$) $\in$ H if there is no r_k such that (r_i, r_k, $v = 0$) $\in$ G and (r_k, r_j, $v = 0$) $\in$ G, or (r_i, r_k, $v = 1$) $\in$ G and (r_k, r_j, $v = 1$) $\in$ G. Similarly for the case (r_i, r_j, $v = 1$). That is, an unmarked link (r_i, r_j) of G is in H whenever there is no read r_k lying "in-between" r_i, r_j with respect to the unmarked subgraph of G.

It is easy to see that in the absence of repeats and sequencing errors, and given enough coverage of the genome by reads, H would simply be a path from the leftmost to the rightmost read in the genome. In particular nodes that have more than two neighbors in H are caused either by repetition, or by sequencing errors.

We define the merge operation on pairs of reads. Reads r_i and r_j can be merged if one of the following conditions holds:

1. (r_i, r_j, $v = 0$) $\in$ H, r_j is the only read extending r_i to the right, and r_i is the only read extending r_j to the left, with respect to H (or vice versa). We say that r_i can be

[1] More precisely, the overlap associated with link (r_i, r_j, v, b_1, e_1, b_2, e_2) is defined to be $\max(e_1-b_1+1, e_2-b_2+1)$.

merged to the right with r_j and r_j can be merged to the left with r_i. (or vice versa), and *with merge bit* = 0 (forward).

2. $(r_i, r_j, v = 1) \in H$, r_j is the only read extending r_i to the right, and r_i is the only read extending r_j to the right, with respect to H (or both extend each other to the left). We say that r_i and r_j can be merged to the left (to the right) with each other, and with *merge bit* = 1 (reverse complement).

Finally sequence contigs are created by merging pairs of reads, by the algorithm below, ensuring that the resulting merged structures are interval graphs in the unmarked subgraph of G. The data structure used to implement sequence contigs is an interval RED-BLACK tree (Cormen/Leiserson/Rivest, 1990).

```
Create a list of reads L.
Do until all reads in L either can be merged to the left and right, or not at all.
    Starting from a read r that can be merged in exactly one direction (left without loss of generality).
        Create new sequence contig S with one element, r.
        Set current direction to left.
        Do until r cannot be merged towards the current direction.
            Find the neighbor r' of r in H, towards the current direction.
            Insert r' in S, and erase it from the list of reads.
            If r, r' can be merged with merge bit = 1, flip current direction (left/right).
            Let r = r'.
```

It is tedious but simple to prove that (1) the above algorithm terminates, and (2) the set of reads in any sequence contig defines a subgraph of the unmarked portion of G, that has the interval graph property.

2.2.4 Assembly of Contigs into Supercontigs

The last stage of *ARACHNE* involves building supercontigs by making use of the forward/reverse linking information between reads that come from the same insert.

As a first step towards this goal, the graph G_C of contig/contig links is computed. A contig f is linked to a contig g if (1) one of the extreme reads of f (leftmost or rightmost) overlaps with Q $< C_{rep}$ one of the extreme reads of g, or (2) at least one read in f is linked to at least one read in g by an earmuff link. If case (1) above holds, we say that f and g are linked with an *overlap* (in addition possibly to being linked by earmuffs). Each link in G_C then has the following associated fields:

1. A bit ovl(f,g) 0/1 indicating the existence of an overlap link.

2. A weight $w(f,g) \geq 0$ indicating the number of earmuff links.

3. A distance d(f,g) between the endpoints of f and g that overlap, taken as the average of estimated distances coming from the earmuff links, and the overlap link if there is one. The distance can be negative, indicating overlap (ovl(f,g) does not necessarily imply $d(f,g) < 0$).

4. An orientation $v(f,g) \in (0, 1, \textit{void})$ indicating whether the earmuff and overlap links unambiguously determine a relative orientation forward (0), reverse complement (0), or if different links disagree (*void*) about the relative orientation between f and g.

5. A conflict count confl(f, v), which will be defined below.

We define the *conflict count* between a pair of linked sequence contigs f, g. Recall from above that d(f,g) indicates the estimated distance between f and g, *i.e.* the estimated gap (if d>0) or overlap (if d<0) between f and g. Say now f is also linked (towards the same direction) with another sequence contig h, with distance d(f,h). Denote by |f| the estimated length of f, and likewise |g| for g, h. Say for the purpose of exposition, that $d(f,g) < d(f,h)$ and $d(f,g) + |g| > d(f,h)$. If these estimates were accurate, then g, h overlap by an amount equal to $l = d(f,g) + |g| - d(f,h)$. Unless l is less than the length of the extreme reads of g, h, and g is linked to h by an overlap link,

this overlap between g and h is inconsistent. We count in such cases a conflict for the edge (f,g), and one for the edge (f,h).

Unfortunately the estimates of lengths of sequence contigs, and more notably of distances between contigs, are not necessarily accurate. For that reason a tolerance is necessary when determining the conflict count of intercontig links. We define a heuristic tolerance function $\tau(d, d', w, w') \geq 0$, in our example above $\tau(d(f,g), d(f,h), w(f,g), w(f,h))$. In general longer distances and smaller weights of links increase the tolerance τ. For brevity, we will also denote $\tau(d(f,g), d(f,h), w(f,g), w(f,h))$ by τ_{fgh}.

Formally we define a *conflict* for a link (f,g) to be a link (f,h) such that $d(f,g) + |g| > d(f,h) + \tau_{fgh}$, and $d(f,h) + |h| > d(f,g) + \tau_{fhg}$, or a link (g,h) such that $|f| + d(f,h) > d(g,h) + \tau_{hfg}$ and $|g| + d(g,h) > d(f,h) + \tau_{hgf}$. The *conflict count* confl(f,g) is the sum of all the conflicts for link (f,g).

We define a supercontig to be a linked list of contigs, where links have an associated estimated distance, which can be negative if the two linked contigs overlap.

The algorithm for creating the supercontigs simply involves defining a priority for links between sequence contigs, and then executing *merge* operations between sequence contigs, or the supercontigs they belong to, according to the defined link priority. A formal definition of the *merge* operation would be somewhat technical and is omitted. Briefly, two supercontigs S, T (each consisting of one, or more sequence contigs) are merged according to link (f,g) between sequence contigs f in S and g in T, only if the result can be a linked list (supercontig) preserving the exact distances within S and T, and the distance d(f,g) of link (f,g) within some tolerance. Figure 3.5 shows two possible merges, and one impossible merge.

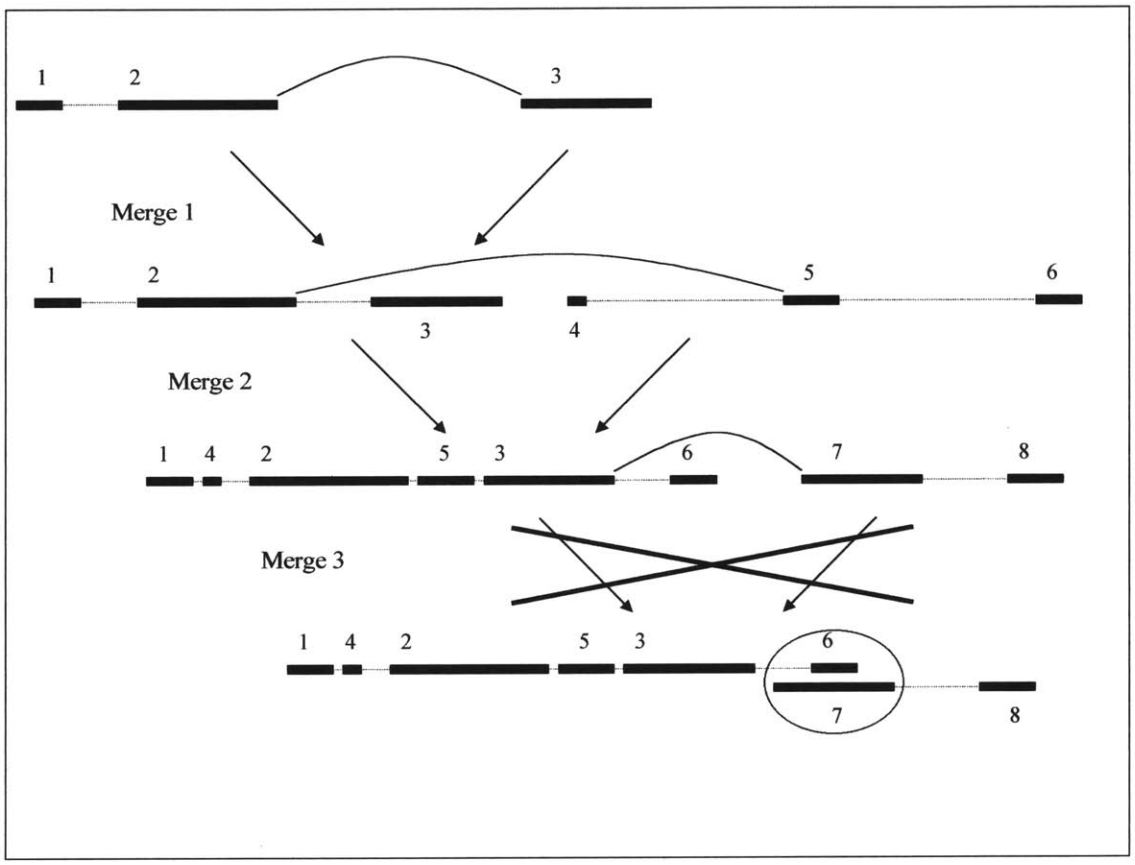

Figure 3.5. Demonstration of three supercontig merges. Sequence contigs 2 and 3 are linked, inducing Merge 1. This merge is possible because contig 3 can be placed to the "right" of contig 2, without overlapping any existing contig to the right of contig 2. Merge 2 is induced by a link between contig 2 and 5. Notice that contigs 2 and 5 are not the rightmost/leftmost contigs of their respective supercontigs. Still, the distance between 2 and 5 is large enough for contig 3 to fit between them, and likewise contig 4 can be placed between contigs 1 and 2 preserving all distances. Merge 3 is induced by a link between contigs 3 and 7. This time the distance associated with the link is not consistent with the presence of contig 6 to the left of contig 3. Merge 3 is not performed.

3. Results

We tested a prototype version of *ARACHNE* on simulated shotgun data that we generated from (1) the complete *Haemophilus influenzae* genome (Fleischmann et al. 1995); (2) the complete *Archaeoglobus fulgidus* genome (Klenk et al. 1997); (3) chromosome I of the *Caenorhabditis elegans* genome (The C. elegans Sequencing Consortium, 1998); (4) chromosome 22 of the

Human genome (Hunt et al. 1999). In this section we report a study on the repeat structure of these genomes as it relates to our hashing scheme for finding read-pair overlaps, and then we report results on the quality of assembly produced by our preliminary implementation.

3.1 Repetition of k-mers in the Test Sequences

Repeats render fragment assembly harder. The most problematic repeats are long, highly preserved segments that are repeated two or more times in the genome. Segments a bit shorter than a typical read length are not a problem, because each particular occurrence is likely to be included in some read, which extends to the flanking unique regions. Such a read virtually resolves the repeat occurrence by making it obvious that the flanking unique regions are next to each other. The rules for discarding repetitive links above target precisely such cases. For instance Alu repeats are shorter than the length of a typical read and they should not pose a major difficulty in the assembly of shotgun reads from human DNA, even though there are around one million occurrences of Alu in the human genome. Repeats that exhibit low sequence similarity among copies are also not a problem, because two reads from the same location on the repeat, but from different repeat occurrences, will not be considered overlapping by the pairwise read alignment procedure.

Sequence	Percent of sequence covered by unique k-mers			
	k = 16	k = 18	k = 20	k = 24
Human chromosome 22	67.8	76.3	79.2	83.8
C. elegans chromosome I	74.5	83.9	87.5	89.6
H. influenzae	94.6	96.0	96.3	96.5
A. fulgidus	96.8	97.9	98.1	98.3

Table 2. Percentage of genomic sequence covered by unique k-mers. This is a measure of the rate of repetition in the sequence. Human sequence is the most repetitive; C. elegans sequence is not much less repetitive. H. influenzae and A. fulgidus are dramatically less repetitive.

We compiled statistics on the composition of the four test sequences. We are specifically looking at k-mers in the sequences, and computing: (1) the distribution of k-mer frequencies in the sequence, and (2) the distribution of *k-repeat-segment* lengths, where a k-repeat-segment is a

maximal contiguous stretch of repeated k-mers. For example k-repeat-segments of length 0 correspond to unique k-mers, while k-repeat segments of length 1 correspond to 1 consecutive repeated k-mers flanked by two unique k-mers.

A k-mer is *unique* if it appears exactly one in the sequence in both forward and reverse complement direction. It is interesting to compare the number of unique k-mers for the different test sequences. Table 2 makes this comparison, for k = 16, 18, 20, and 24. It is clear from this table that Human chromosome 22 exhibits the highest rate of repetition of k-mers, C. elegans has slightly lower repetition, while H. influenzae and A. fulgidus have dramatically lower rates of repetition.

Figures 6, 7, 8, and 9 show the distribution of k-mer frequencies in the four test sequences. The x-axis corresponds to frequency, and the y-axis corresponds to $\log_{10}(F(x)+1)$ where F(x) is the number of distinct 24-mers that occur exactly x times each in the sequence, in the forward or reverse complement direction.

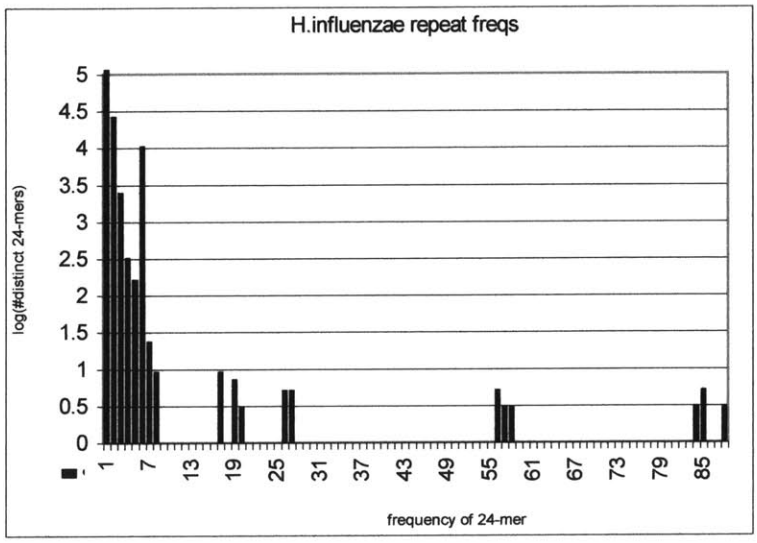

Figure 3.6. Number of distinct 24-mers that occur in the H. influenzae genome at specific frequencies.

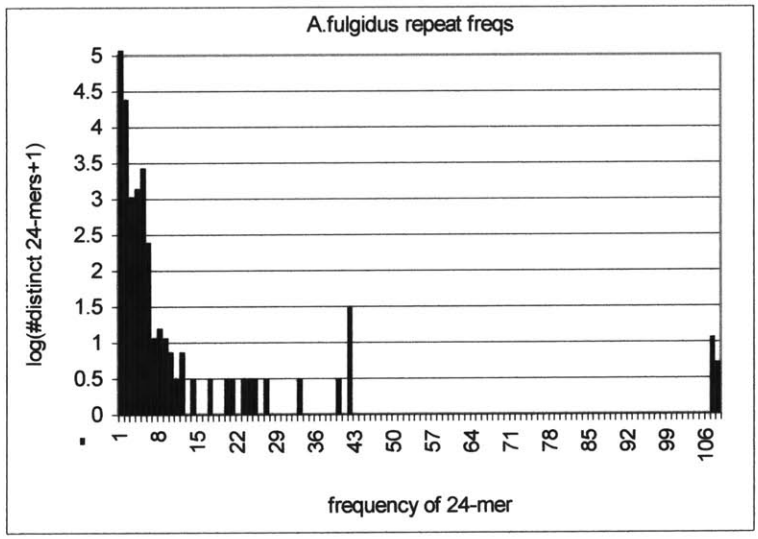

Figure 3.7. Number of distinct 24-mers that occur in the A. fulgidus genome at specific frequencies.

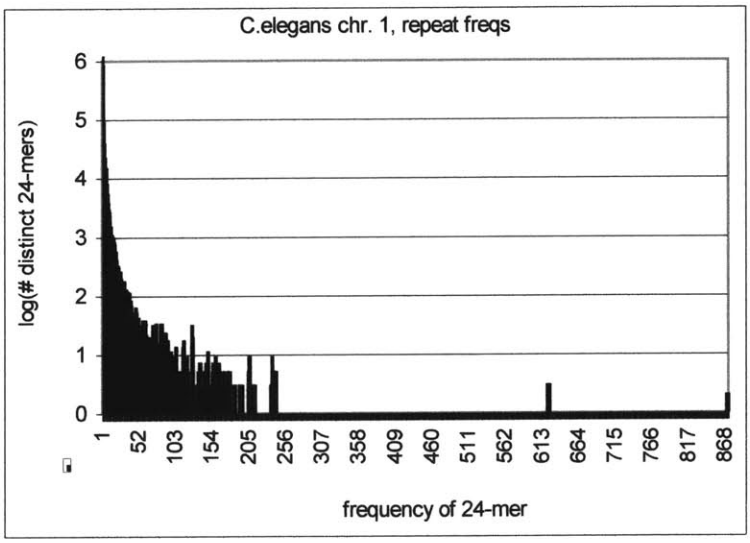

Figure 3.8. Number of distinct 24-mers that occur in C. elegans chromosome I at specific frequencies.

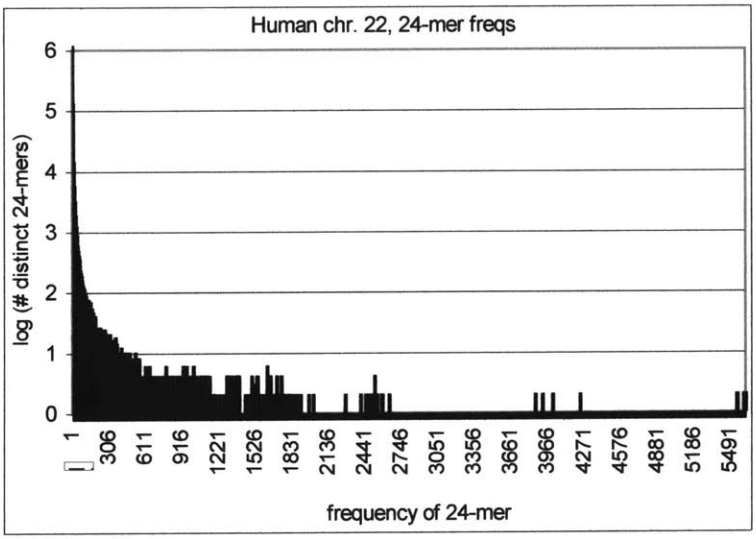

Figure 3.9. Number of distinct 24-mers that occur in human chromosome 22 at specific frequencies.

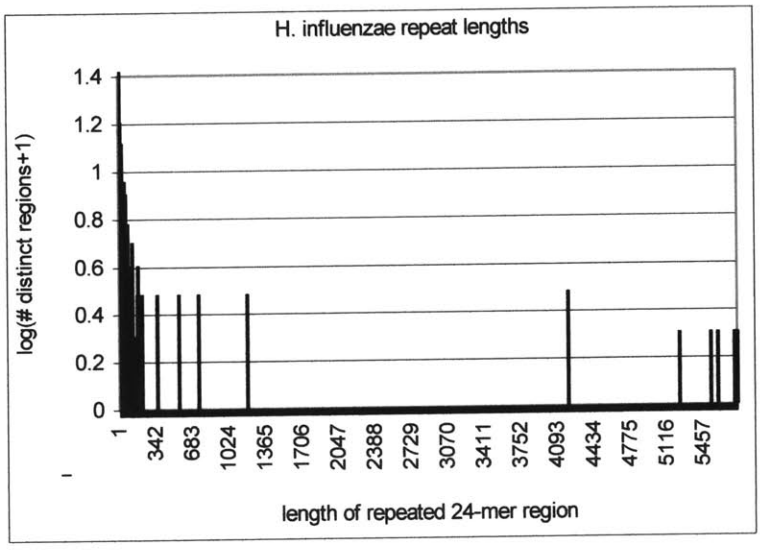

Figure 3.10. Lengths of repeat 24-mer regions in the H. influenzae genome.

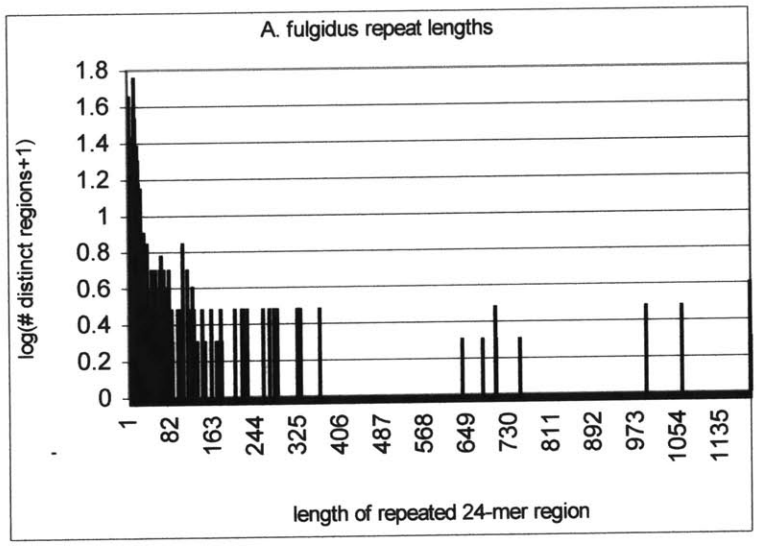

Figure 3.11. Lengths of repeat 24-mer regions in the A. fulgidus genome.

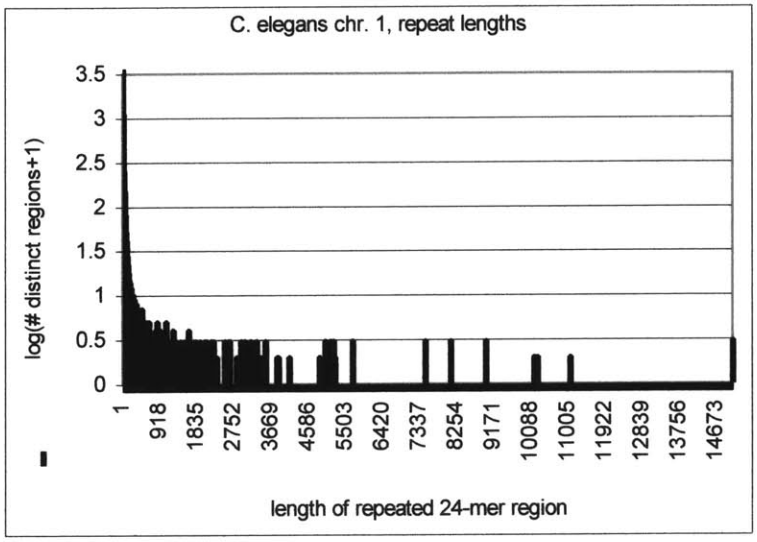

Figure 3.12. Lengths of repeat 24-mer regions in chromosome 1 of C. elegans

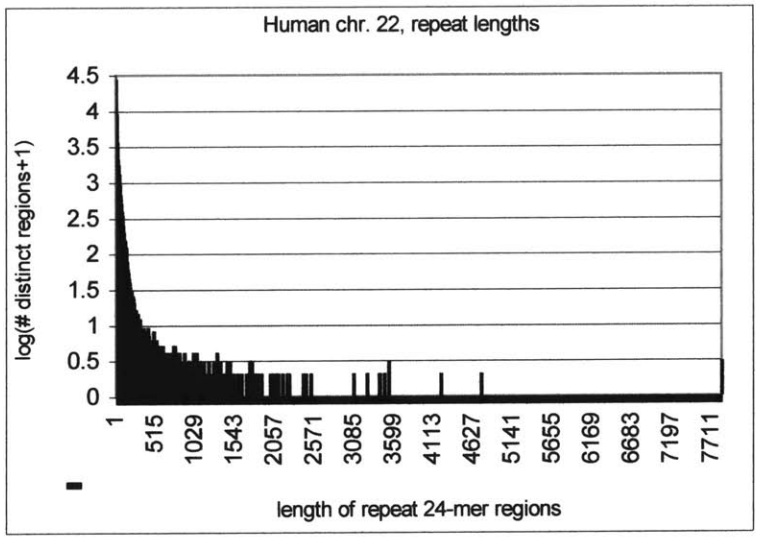

Figure 3.13. Lengths of repeat 24-mer regions in chromosome 22 of the Human genome.

Figures 10, 11, 12, and 13 show the distribution of lengths of uninterrupted stretches of repeated 24-mers. The x-axis corresponds to number of consecutive repeated 24-mers. The y-axis corresponds to $\log_{10}(G(x)+1)$ where G(x) is the number of distinct regions that consist of x consecutive repeated 24-mers flanked by two unique 24-mers.

The distributions of repeat frequencies of H. influenzae and A. fulgidus look very similar. In both genomes, most repetitive 24-mers are repeated very few times, generally less than 10 times. A few 24-mers are repeated up to 100 times. In H. influenzae one can observe a peak of 24-mer occurrences at 6. The reason is a repeat of length around 6,000 that occurs 6 times. This can be observed by looking at the repeat lengths distribution.

The distributions for the human and C. elegans chromosomes are very different from the ones above. One interesting observation is that although the human chromosome is more repetitive (it has a smaller percentage of unique k-mers as we saw above), the C. elegans genome seems to have better preserved repeats. There are several regions of length above 5000 that consist of only repeated 24-mers in the C. elegans chromosome I. The picture in the human chromosome 22 is considerably different, with much fewer long regions of repeated 24-mers. This observation may be significant for shotgun assembly. If two repeat occurrences are dissimilar enough, the pairwise alignment algorithm will not connect two reads that come from different occurrences. Therefore if repeats are better preserved in C. elegans than in human, then shotgun assembly on the human genome may actually be easier than in C. elegans.

3.2 Generation of Shotgun Data

There were very few unknown nucleotides, and a few gaps in the test sequences. We removed all those by gluing the flanking regions together. We believe this step affects minimally the sequences for the purpose of testing the *ARACHNE* prototype implementation. In real data there should not be any unknown nucleotides. Instead, real data have confidence scores (*PHRED* scores) for each nucleotide.

In all test cases we used the shotgun assembly conditions below:

1. All reads come in earmuff links.

2. A third of the earmuff links are 10,000 *long*, and the rest are 3,000 *long*.

3. Earmuff link distances have an average error of 5%, under Poisson distribution.

4. Reads come from either orientation, forward or reverse complement, with equal probability. Two linked reads come always from the same orientation.

5. Sequencing errors are distributed according to Poisson processes in each read. The average error rate is 0.5% for mutations, 0.05% for insertions, and 0.05% for deletions.

The above error rates are in the lowest range of the realistic spectrum. Tuning of the algorithms and parameters depends on the error rates. For example, if there are no errors in sequencing, then two overlapping reads should be required to agree perfectly on the area of overlap. As sequencing error rates increase, the pairwise alignments procedure needs to be tuned accordingly. As *ARACHNE* is in a prototype phase, and because of limited disk space/CPU time we did not perform additional tests with different error rates, and/or different parameters.

3.3 Fragment Assembly Results

We report performance in terms of how much of the genomic sequence is covered with long supercontigs, and how "correctly" those supercontigs are assembled.

Thus we look at all supercontigs above a cutoff *estimated length*. Estimated length of the supercontig is the length in nucleotides that the algorithm estimates for the supercontig. This measure includes the sum of lengths of all contigs plus the sum of estimated lengths of gaps in between the contigs of the supercontig.

We define the true position of a contig by taking a majority vote among all reads of the contig. Each read has one "vote" for the leftmost and rightmost positions of the contig on the genome, depending on the true location of the read on the genome (which we know). If a majority of the reads agrees within a few nucleotides to a certain position, then these reads are

considered *good*, and the rest are considered *misplaced*. If there is no majority agreement, the whole contig is considered *misassembled*, and the reads of it misplaced.

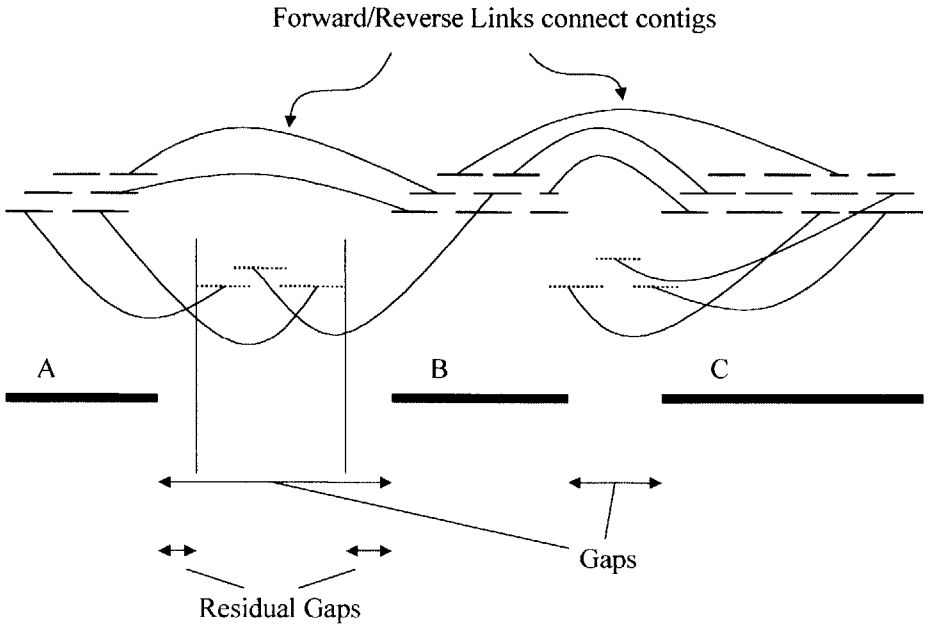

Figure 3.14. A supercontig consisting of three contigs. Gaps and residual gaps between the contigs are illustrated.

We similarly define the true position of a supercontig by a majority vote among the contigs. We consider a contig *good* if it agrees with the majority vote within a slack factor of 1% of the supercontig length, otherwise we consider the contig *misplaced*.

We measure the following quantities:

1. *Genome covered.* This quantity measures how many positions as a fraction of total genome positions, are covered by all supercontigs above the cutoff length. A position is

covered if it lies within the range of a good contig, or within the gap range between two neighboring good contigs in the same supercontig.

2. *Percentage of contigs that are misplaced.* Percentage of contigs that are misplaced within the supercontigs above the cutoff length.

3. *Percentage of all reads that are misplaced.* Percentage of reads that are misplaced within any contig belonging to a supercontig above the cutoff length.

4. *Percentage of reads misplaced inside good contigs.* Percentage of reads that are misplaced within only the good contigs belonging to a supercontig above the cutoff length.

5. *Percentage of gap length.* Percentage of the length of supercontigs that is in gaps between neighboring contigs (Refer to Figure 3.14).

6. *Percentage of residual gap length.* Percentage of the length of supercontigs that is in gaps between neighboring contigs, and is not covered by reads linked with an earmuff to any good read assigned to a contig inside the supercontig (Refer to Figure 3.14).

7. *Percentage of unassigned reads.* Percentage of reads completely included within the true boundaries of contigs (excluding misassembled contigs) inside the long supercontigs, not actually placed within those contigs.

We report statistics for supercontigs longer than 1Mb, 250kb, 100kb, 25kb, 10kb, 5kb, 2kb, and 1kb.

Figure 3.15 shows the percentage of genome covered by contigs above the cutoff lengths, on the following test runs: (1) human chromosome 22, 9x coverage (HUM 9x); (2) C. elegans chromosome 1, 9x coverage (CEL 9x); (3) H. influenzae, 9x coverage (HIN 7x); (4) A. fulgidus, 7x coverage (AFL 7x).[1] We observe that the A. fulgidus genome is virtually perfectly covered

[1] We provide data on tests with lower coverage on the small genomes, because in general these genomes are considered easier to sequence, and thus it may be optimal to use lower coverage with shotgun reads for such genomes.

with a single supercontig (99.96%), while the H. influenzae genome is similarly covered to 99.6% with one supercontig > 1Mb and another > 200K long. The C. elegans genome is covered to a good extent (97.5%) with supercontigs > 25K long, while the human genome is covered 95% with supercontigs > 10K.

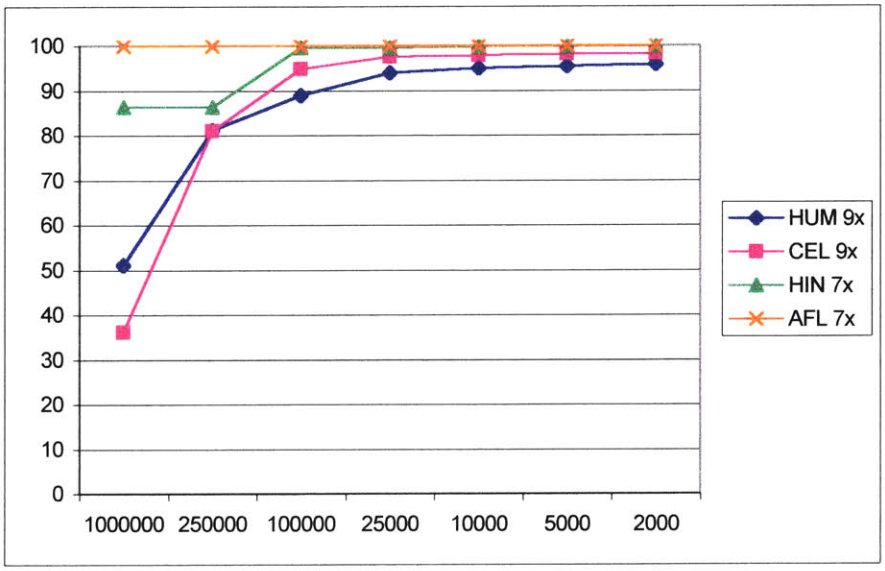

Figure 3.15. Coverage of genomes by long supercontigs.

For each of the above tests we plotted statistics 2-7 in order to measure the quality of the assembled supercontigs. Figure 3.16 shows the results for the human chromosome 22, 9x coverage.

CERR is the percentage of contigs that are misplaced; RERR is the percentage of all reads that are misplaced; RERR2 is the percentage of reads that are misplaced inside good contigs; GAP is the percentage of gap length; RGAP is the percentage of residual gap length; ROUT is the percentage of unassigned reads.

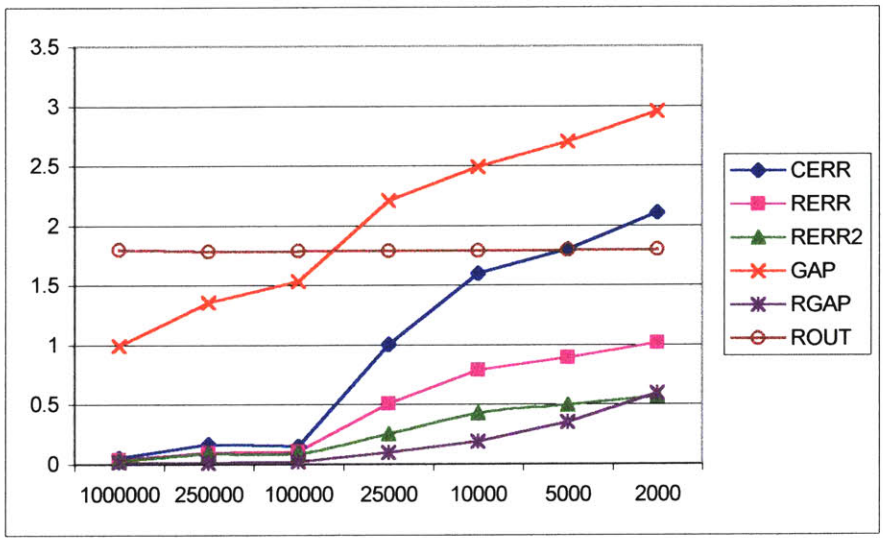

Figure 3.16. Quality of shotgun assembly on the human chromosome 22, 9x coverage with shotgun reads.

Considering GAP and RGAP, we see that gaps inside supercontigs are fairly short, and almost all of them are covered with reads that are linked with earmuffs to the proximal contigs. This fact should make the assembly of sequence that lies in gaps between contigs a fairly easy task in a subsequent version of *ARACHNE*.

Considering ROUT, this quantity can be thought of as reads that are "wasted" because they are not assigned to the place in the assembly where they belong. Since it is below 2% of the total reads, we believe it is a fairly small wastage.

Finally the most difficult statistics to evaluate are CERR, RERR, and RERR2, relating to misassembled data. For the long supercontigs, these rates are extremely low, while for smaller supercontigs the rates get fairly high. In general these rates reflect regions of low copy repetition, where reads from two virtually identical places in the genome are connected into a single contig. Therefore the sequence that the assembled reads represent should for the most part be correct. Moreover there is the possibility of correcting some of these errors by use of earmuff link information to verify assembly inside contigs, and with *PHRED* scores that will increase the specificity of detected overlaps between reads.

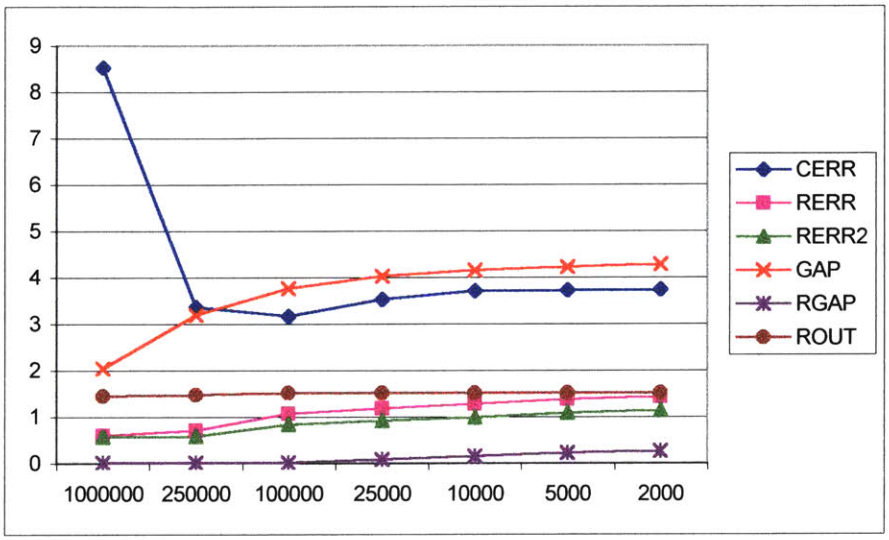

Figure 3.17. Quality of shotgun assembly on C. elegans chromosome I, 9x coverage with shotgun reads.

Figure 3.17 shows the same data for C. elegans chromosome I, 9x coverage. The results are for the most part fairly similar to those for the human chromosome, with one notable exception: the rate of CERR for very long supercontigs is considerably high. This comes from a long supercontig in the assembly, where a highly repetitive region is assembled with most contigs having reads from two places in the genome. It reflects the fact that C. elegans has extremely long, very well conserved low-copy repeats. Incidentally an 11x coverage test on the same chromosome does not have this problem: the CERR rate increases from 0.4% for supercontigs > 1,000,000 upto 2.3% for all supercontigs. No other test run we did has any close to the CERR rate observed in the C. elegans chromosome I, 9x coverage test run.

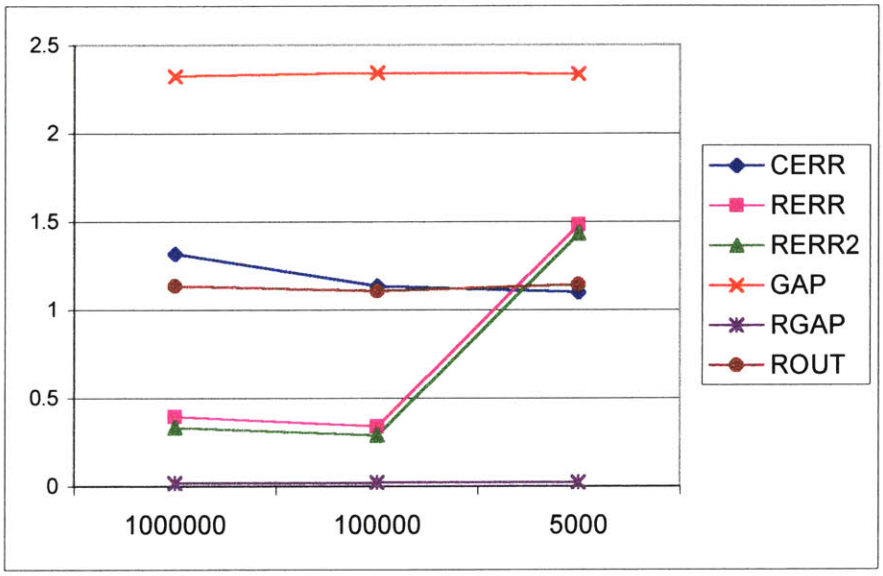

Figure 3.18. Quality of shotgun assembly on the H. influenzae genome, 7x coverage with shotgun reads.

Figure 3.18 shows the same data for the H. influenzae genome, 7x coverage. Note that virtually all of the genome is covered with supercontigs > 100K long. For these supercontigs, the error rates CERR, RERR, and RERR2 are fairly low. Also, RGAP is extremely low and ROUT is fairly low. We believe that the produced assembly for this genome is of very high quality.

Finally, Figure 3.19 shows the data for A. fulgidus, 7x coverage. In this case all the relevant statistics are extremely good, and we believe that this particular genome is assembled fairly easily by our system. Moreover we observe that 7x coverage may be sufficient to assemble genomes that are expected to look similar to H. influenzae and A. fulgidus in terms of size and repetition.

Overall we believe that using techniques similar to *PHRAP* one should be able to provide good consensus sequence of the assembled supercontigs, with fairly low shotgun sequencing wastage. Once these supercontigs are assembled, longer earmuff links (for example coming from BAC clones) should enable mapping the supercontigs with respect to each other resulting in the complete source sequence.

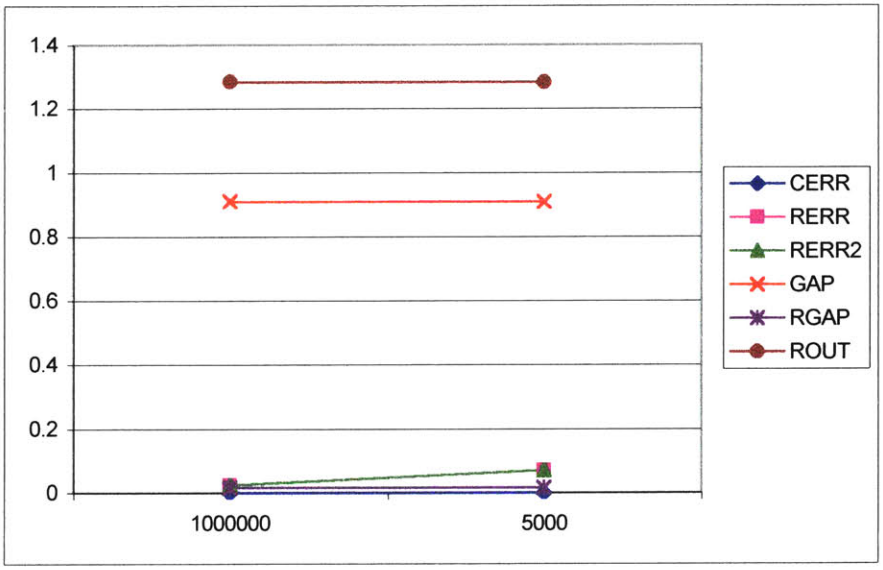

Figure 3.19. Quality of shotgun assembly on the A. fulgidus genome, 7x coverage with shotgun reads.

In Appendix B we provide the tables of statistics, and figures, for all the test runs we performed on the four test sequences.

4 Conclusions and Future Work

We designed, implemented, and tested *ARACHNE*, a system for assembling shotgun sequencing data. To our knowledge the description above is the first published description of such a system, working on real data. We demonstrated good performance of our prototype version of *ARACHNE* on real data coming from the human and C. elegans genomes, indicating that shotgun sequencing assembly is feasible on the challenging human genome. Our results are encouraging that a full finished implementation of *ARACHNE* will succeed in providing good quality assemblies of mammalian genomes.

There is a substantial amount of additional algorithmic and software development needed before we can provide a finished system ready for application on real data. Below we outline some of the steps that need to be taken to reach that goal:

1. Our system as is does not accept real data because of the integration of *ARACHNE* with the procedures generating the simulation data, and the lack of sufficient input/output modules.

2. The implementation was done on a single Alpha processor machine with 2Gb memory and 50Gb storage. Currently this implementation can comfortably handle genomes of size up to 50Mbases with 10x shotgun coverage. The system needs to be redesigned/optimized for a much larger architecture, in order to be applied to a full shotgun sequencing assembly of a mammalian genome.

3. Several parts of the program may not be optimally written, since the development and design were performed concurrently. Thus the system needs to be looked at carefully, re-implementing and optimizing certain parts.

4. *PHRED* scores should be taken into account in the final version, in order to take advantage of the additional information and possibly distinguish much better between true read/read overlaps, and overlaps induced by repeats.

5. Closing gaps between contigs inside a supercontig, by using shortest paths in the graph of read overlaps, between the leftmost/rightmost reads in the contigs, and by making use of earmuffs that connect reads in the gap with reads in the flanking contigs.

6. Backtracking. Earmuff links and *PHRED* scores may be helpful in detecting and correcting the few reads and contigs that are misplaced in the current implementation.

7. Integration with *PHRAP*, or implementation of much of *PHRAP*'s techniques so that the final output is consensus sequence, rather than a layout of reads.

Development towards the above directions is currently under way.

Chapter 4

CROSS-SPECIES GENOMIC COMPARISON AND GENE RECOGNITION

1. Introduction

A fundamental task in analyzing genomes is to identify the genes. This is relatively straightforward for organisms with compact genomes (such as bacteria, yeast, flies and worms), because exons tend to be large and the introns are either non-existent or tend to be short. The challenge is much greater for large genomes (such as those of mammals and higher plants), because the exonic 'signal' is scattered in a vast sea of non-genic 'noise'. Whereas coding sequences comprise 75% of the yeast genome, they represent only about 3% of the human genome.

Computational approaches have been developed for gene recognition in large genomes, with most employing various statistical tools to identify likely splice sites and to detect tell-tale differences in sequence composition between coding and non-coding DNA (Burset and Guigo, 1996). Some programs perform "de novo" recognition, in that they directly use only information about the input sequence itself. One of the best programs of this sort is GENSCAN (Burge 1997), which employs a Hidden Markov Model approach to scan large genomic sequences. Other programs employ "homology" approaches, in which exons are identified by comparing conceptual translation of DNA sequence to databases of known protein sequences (see, e.g., Pachter et al. 1999). An example is the Procrustes program (Gelfand et al. 1996).

In this chapter, we explore a powerful, new approach to gene recognition by using cross-species sequence comparison — that is, by simultaneously analyzing homologous loci from two related species. Specifically, we focus on the ability to accurately identify coding exons by comparison of syntenic human and mouse genomic sequence.

It is well known that cross-species sequence comparison can help highlight important functional elements such as exons, because such elements tend to be more strongly conserved by evolution than random genomic sequence. If a protein encoded by a gene is already known in one organism, it is relatively simple to search genomic DNA from another organism to identify genes encoding a similar protein (using such computer packages such as Wise2, http://www.sanger.ac.uk/Software/Wise2/). A more challenging problem is to identify exons directly from cross-species comparison of genomic DNA. Computer programs are available that identify regions of sequence conservation (Jang et al. 1999), which can then be individually analyzed in an ad hoc fashion to see whether they may contain such features as exons or regulatory elements. Preliminary work on human gene recognition using comparison with the *tetraodondiform fugu rubripes* genome has been reported (H. Roest Crollius et al. 1999). In that work, 13 homologous genes from fugu and human with > 40% sequence similarity were selected out of a set of 17 human/fugu homologs. These genes were used in order to implement and test a procedure based on *TBLASTX* (http://www.ncbi.nlm.nih.gov/BLAST/) sequence comparisons on the amino acid level, detecting exons in human DNA. The procedure was further tested on sequence data from another tetraodon organism, *tetraodon nigroviridis*. Preliminary results suggested that human exons may be detectable with >95% specificity (H. Roest Crollius et al. 1999). However, there has been no systematic reported approach to employing cross-species comparison for gene recognition.

We developed an automatic approach to exon recognition, by using cross-species sequence comparison to identify and align relevant regions and then searching for the presence of exonic features at corresponding positions in both species. We began by undertaking a systematic comparison of the genomic structure of 117 orthologous gene pairs from human and mouse, to understand the extent of conservation of the number, length and sequence of exons and introns. We then used these results to develop algorithms for cross-species gene recognition, consisting of *GLASS*, a new alignment program to provide good global alignments of large genomic regions by using a hierarchical alignment approach, and *ROSETTA*, a program to identify coding exons in both species based on coincidence of

genomic structure (splice sites, exon number, exon length, coding frame, and sequence similarity).

ROSETTA performed extremely well in identifying coding exons, showing 95% sensitivity and 97% specificity at the nucleotide level. The performance was superior to programs that use much more sophisticated signals and statistical analysis but analyze only a single genome (Burset and Guigo 1996, Burge 1997). To our knowledge, *ROSETTA* is the first program for gene recognition based on cross-species comparison. The approach can be readily generalized to other pairs of organisms, as well as to the study of three or more organisms simultaneously.

With the current explosion of human and mouse genomic sequence, cross-species comparison is likely to provide one of the most powerful approaches for extracting the information in mammalian genomes.

2. Results

2.1 Comparison of Human and Mouse Genomic Loci

Comparison of mRNA sequences of 1196 orthologous human and mouse gene pairs were recently reported (Makalowski et al. 1996), showing that coding regions tend to show roughly 85% identity at the nucleotide and protein levels. We sought to extend this analysis by comparing genomic structure, where known. The mRNA sequences from the orthologous gene pairs were searched against GenBank Release 109 (October 1998), to identify those for which genomic sequence was available in both species. Entries were required to contain the complete genomic sequence encompassing all coding exons, although not necessarily including the introns between non-coding exons.

A total of 117 orthologous gene pairs were identified and studied (Appendix C, Table C.1). For the purpose of comparing the genomic structure of the gene pairs, we used dynamic programming algorithms (employing both nucleotide similarity and codon similarity using the PAM20 matrix (Dayhoff et al. 1978) to align the sequences and carefully inspected the alignments to ensure that they correctly aligned the exons.

The comparison defined the striking extent of evolutionary conservation:

(i) *Exon number*. The number of exons was identical for 95% of the genes studied. There were six instances in which the number of exons differed.

In two cases, a single internal coding exon in mouse is reported to correspond to two internal coding exons in human. In the spermidine synthase gene (Appendix C, Table C.1, gene 30), mouse exon 5 corresponds to human exons 5 and 6, with the total exonic lengths agreeing perfectly. In the lymphotoxin beta gene (Appendix C, Table C.1, gene 85), mouse exon 2 corresponds to human exons 2 and 3. Interestingly, the mouse exon 2 is 316 bp while the sum of the lengths of human exon 2, intron 2 and exon 3 is only 301 bp.

In the next three cases, the correspondence broke down for terminal exons. In the keratin 13 gene (Appendix C, Table C.1, gene 40) and the adenosine deaminase gene (Appendix C, Table C.1, gene 66), the coding sequences show substantial sequence divergence at the 3'-end and one of the organisms has an extra exon. In the proteosome LMP2 gene (Appendix C, Table C.1, gene 46), the extra human exon shows striking sequence similarity to a portion of the 3'-UTR in the mouse.

The final case was also in the LMP2 gene, in which the first two coding exons in the human correspond to one exon in the mouse. There is no apparent relationship between their lengths (even including the intron).

It is possible that some of the apparent differences are due to error in annotation in the databases.

(ii) *Exon Length*. The length of corresponding exons was strongly conserved. The lengths were identical in 73% of cases. Those differences that did occur were quite small: the mean ratio of the larger to smaller length was 1.05.

Moreover, the differences were nearly always a multiple of three. The length difference was a multiple of three for 95% of all exons and 99% of all internal coding exons. This is readily understood in terms of the effects of evolutionary selection: length differences divisible by three alter an integral number of codons, while other length differences would require a second compensatory change in a succeeding exon to restore the translational reading frame and would thus be less likely.

Only three instances were found in which corresponding internal exons had lengths differing by other than a multiple of three.

In the skeletal muscle specific myogenic gene (Appendix C, Table C.1, gene 49), the respective lengths of exons 2 and 3 are 81 bp and 123 bp in the human and 82 and 122 in the mouse. Remarkably, two instances occur in the gene encoding the Flt3 ligand (Appendix C, Table C.1, gene 100). The respective lengths of exons 2 and 3 are 111 bp and 54 bp in the human and 122 bp and 46 bp in the mouse, while the respective lengths of exons 5 and 6 are 139 bp and 179 bp in the human and 144 bp and 189 bp in the mouse.

(iii) *Intron Lengths*. Whereas exon lengths tended to be well preserved, intron lengths varied considerably. The mean ratio of the larger to the small length was 1.5. As would be expected, there was no tendency for intron lengths to differ by a multiple of three. Human introns tended to be larger than mouse introns (68% of cases), but this could represent a selection bias reflecting the fact that the less extensive sequencing of the mouse genome may lead to an underrepresentation of instances in which the mouse genomic locus is larger. This question will need to be revisited in the presence of larger amounts of genomic sequence.

(iv) *Sequence Similarity*. Coding regions showed strong sequence similarity, with roughly 85% identity as previously reported (Makalowski et al. 1996, 1998). In contrast, introns showed only weak sequence similarity with roughly 35% sequence identity — which is not much higher than the background rate of sequence identity in gapped alignment of random sequence.

The degree of conservation varied considerably among genes. For example, the gene encoding the ribosomal protein S24 (Appendix C, Table C.1, gene 4) showed 88% identity at the DNA level and 100% identity at the amino acid level in coding exons, but only 27% identity at the DNA level in introns. The perfect identity at the amino acid level is consistent with the protein being highly constrained, as might be expected for a component of the ribosome. In contrast, some introns exhibited a striking degree of similarity. In the tumor necrosis factor-beta gene (Appendix C, Table 1, gene 93), the first intron has 75% nucleotide identity and nearly perfect agreement in length (86 bp in human, 83 bp in mouse). Interestingly, the flanking exons are less well conserved, showing only 70% nucleotide identity and 60% amino acid identity.

2.2 Global Sequence Alignment, GLASS

To recognize genes based on the coincidence of biological signals in two organisms, it is important to start with an accurate global alignment of the genomic sequences. Existing global alignment techniques were not well suited for our purposes:

1. Standard dynamic programming methods (based on the Needleman and Wunch (1970) or Smith and Waterman (1981) algorithms) were unsuitable for two reasons. First of all, these methods are too slow because they involve computing an O(NM)-size dynamic programming matrix of alignment between the two input sequences of lengths N and M. For our purposes N and M can be quite large. In our database the largest entries are longer than 30kb. Moreover we would like our programs to be usable on sequences that are much longer such as large-insert BAC and YAC clones, or larger contigs, that can range from 150k to several megabases. Secondly, these standard methods are not sensitive to finding short regions of good alignment, flanked by much longer regions of poor alignment. For our purposes, we would like to be able to detect regions of good alignment (exons) that can be as short as 30-50 bases, flanked in the human and mouse sequences by much longer regions (introns) that are poorly conserved and have vastly different lengths.
2. Faster local alignment methods such as *BLAST* (http://www.ncbi.nlm.nih.gov/BLAST/) are better suited for our purposes, but still insufficient. One reason is that *BLAST* does not provide a global map of the two input sequences. Instead it provides a set of local alignments, ranked by quality. One approach then would be to try to use *BLAST* in order to detect these local alignments, and then arrange the best ones into an ordered set of local maps that can be extended to a global alignment. But another problem with *BLAST* is that it uses perfect matches between k-long words of a fixed size k in order to detect local alignments between the two sequences. The default value for k is 11, and this value is very large for the purpose of detecting short exons that may not have an 11-long

matching word between the human and mouse. On the other hand using a smaller value for k leads to detecting too many local alignments.

Accordingly, we designed a new alignment system called *GLASS* (GLobal Alignment SyStem), suitable for aligning hundreds of kilobases of genomic sequence. *GLASS* works by iteratively aligning matching segments (Figure 4.1). First, a rough alignment map is constructed by finding long segments that match exactly, and whose flanking regions have high similarity. The procedure is repeated on the intervening regions using successive smaller matching segments. Finally, the remaining short unaligned regions are aligned using standard alignment techniques.

More precisely, *GLASS* works as follows. The program takes as input two genomic segments and returns a global alignment for the segments. The global alignment is computed recursively. The basic steps are as follows:

1. For an initial value of k, find all matching k-mers in the pair of sequences, i.e. all k-mers that appear in both sequences.
2. Treating each matching k-mer as a unique "character", convert both the human and mouse sequences into sequences of such characters, corresponding to occurrences of the matching k-mers, temporarily leaving out unmatched k-mers.
3. Align these two strings using standard dynamic programming with the following caveat. Matching k-mers receive a score equal to the sums of the alignment scores obtained by standard dynamic programming on the short regions flanking the occurrence of the k-mer in the human and mouse sequence (specifically the 12 bases to the left and 12 bases to the right). On the short flanking regions, the standard dynamic programming is run with scores for match, mismatch, gap open, and gap extension of 1, -1, -3, and –2 respectively. Henceforth, these will be the default parameters when we run standard dynamic programming on nucleotides. Mismatches and gaps in the k-mer alignment receive a score of 0.

4. In the above alignment, identify those pairs of matching k-mers that lie within regions of good local alignment between the human and mouse sequences — that is, that have a score exceeding a threshold T (typically $0<T<6$).
5. From this list of pairs of matching k-mers, remove those that are inconsistent with the underlying human and mouse genomic sequences. Specifically, two k-mers are inconsistent if they correspond to positions that overlap by $i>0$ bases in the one species but not in the other species.
6. Using the remaining list of matching k-mers, fix the alignment between the nucleotides in the underlying human and mouse sequences contained in these k-mers.
7. Recursively align the regions between aligned nucleotides, by repeating steps 1-6 using a smaller value of k. As currently implemented, *GLASS* recursively employs k-mers with k = 20, 15, 12, 9, 8, 7, 6 and 5.
8. Once the last recursive alignment is performed, extend all pairs of aligned segments by short (of length 12) local alignments to the left and right by SDP.
9. Finally, align the remaining (usually short) unaligned regions using SDP.

Various parameters used in the *GLASS* program were adjusted on the basis of a test set consisting of 12 orthologous gene pairs (Table 4.1). An example of an alignment between two orthologous genomic loci is shown in Figure 4.1.

Once the genomic sequences are aligned, the sequences are processed to mask repeat elements (using the RepeatMasker program) and poorly aligned regions (defined as those containing too many gaps or too many mismatches). The remaining, well-aligned sequence was then used for gene recognition.

MUMmer is another system that provides global alignment of similar sequences, which was developed independently of GLASS (Delcher et al. 1999). *MUMmer* however does not provide the entire alignment map between regions of good alignment, and moreover it uses more relaxed criteria than steps 3-5 of our algorithm above for fixing parts of the global alignment based on long matching k-mers. However *MUMmer*'s implementation runs faster than our implementation of *GLASS*.

2.3 Gene Recognition, ROSETTA

To perform gene recognition, we began by specifying the 'gene model' to be recognized in a genomic region.

A "coding exon" was defined to be the translated portion of an exon, together with a designated strand and reading frame. Coding exons can be "initial" (consisting of the region from the start of translation to either a splice site or the in-frame stop codon), "internal" (consisting of the region between two splice sites), or "terminal" (consisting of the region from either a splice site or a start of translation to the in-frame stop codon). Coding exons thus differ from actual exons in that they exclude the nucleotides in the 5'- and 3'-UTRs.

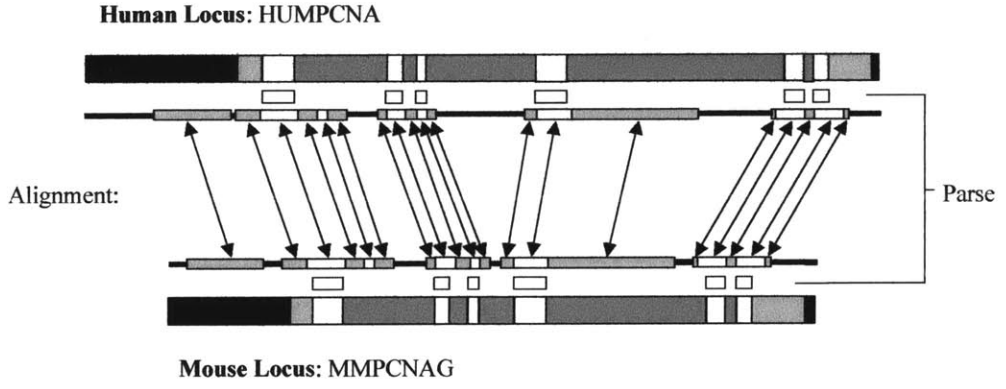

Figure 4.1: Regions of the human and mouse homologous genes: Coding Exons (*white*), Noncoding Exons (*gray*), Introns (*dark gray*), and intergenic regions (*black*). Corresponding strong (*white*) and weak (*gray*) alignment regions of GLASS are shown connected with arrows. *Dark lines* connecting the alignment regions denote very weak, or no alignment. The predicted coding regions of ROSETTA in human, and the corresponding regions in mouse, are shown (*white*) between the genes and the alignment regions.

A "parse" of a genomic region is a sequence $((a_1, b_1, t_1, s_1, f_1), ((a_2, b_2, t_2, s_2, f_2), \ldots ((a_n, b_n, t_n, s_n, f_n))$ where $e_i = (a_i, b_i, t_i, s_i, f_i))$ denote consecutive exons with starting and stopping points (a_i, b_i), type $t_i \in$ {initial, internal, terminal}, designated strand $s_i \in \{+, -\}$ and reading frame $f_i \in \{0,1,2\}$. The parse is "valid" provided that the following properties hold for each pair of consecutive coding exons e_i and e_{i+1}: (i) If e_i is terminal, then e_{i+1} is initial, and vice versa, and (ii) if e_i is not terminal, then e_i and e_{i+1} have consistent strands and reading frames and are both open in the designated reading frame.

Currently strands are handled separately, and parses in the two strands are merged in a post-processing step.

For details we refer to our website (http://theory.lcs.mit.edu/crosspecies/).

Our automatic procedure involved using a dynamic programming approach to find the optimal valid parse with respect to a given scoring procedure. Each parse $((a_1, b_1, t_1, s_1, f_1), ((a_2, b_2, t_2, s_2, f_2), \ldots ((a_n, b_n, t_n, s_n, f_n))$ of the human genomic sequence corresponds to a parse $((a'_1, b'_1, t'_1, s'_1, f'_1), ((a'_2, b'_2, t'_2, s'_2, f'_2), \ldots ((a'_n, b'_n, t'_n, s'_n, f'_n))$ of the mouse genomic sequence, by means of the cross-species sequence alignment. Each parse is assigned a score consisting of the sum of scores for the individual coding exons. The score for each coding exon consisted of several components, reflecting the presence of appropriate splice sites, codon usage, amino acid alignment and length.

(i) Splice Sites. Splice site scores were calculated by using a hybrid method that combines the GENSCAN splice site detector (Burge, 1997) and a directionality effect (Pachter, 1999, Pachter et al. 1999). The splice site scores for the splice acceptor and splice donor sites in both the human and mouse sequence were summed to obtain an overall score for each putative coding exon. (For initial or terminal exons, splice site scores were only computed at the appropriate end.)

(ii) Codon Usage. Codon usage score was computed for both the human and mouse exons, and the two scores were added together. Each score was calculated by summing the log odds ratio for each codon, based on published codon frequencies for the organism (Delphin, M., et al. 1999).

(iii) Amino Acid Similarity. An amino acid similarity score was calculated by comparing corresponding codons in the two exons and using the PAM20 matrix to score

matches, mismatches and gaps. This score reflected the tendency of particular amino acid substitutions to occur between human and mouse (Dayhoff et al. 1978).

1. (HUMIL5A, MMIL5G)	7. (HUMAPEXN, MUSAPEX)
2. (HUMCAPG, MUSCATHG)	8. (HUMERPA, MUSERPA)
3. (HUMSMPD1G, MMASM1G)	9. (HUMVPNP, MUSVASNEU)
4. (HSHOX3D, MMU28071)	10. (HUMIL9A, MUSP40M)
5. (HUMTRPY1B, MUSPROT6A)	11. (HSFAU1, MUSFAUA)
6. (D67013, MMAJ2146)	12. (HUMTHY1A, MUSTHY1GC)

Table 4.1. Training set of human/mouse homologs.

(iv) Exon Length. An exon length score was calculated, consisting of two components. The first component reflected agreement with the known length distribution of initial, internal and terminal exons. The second component penalized exons pairs that differed in length, particularly when the difference was not a multiple of 3.

Various parameters were optimized, based on analysis of the test set of 12 orthologous genes (Table 4.1). The precise definitions of the scores are available on our web site (http://theory.lcs.mit.edu/crosspecies/).

2.4 Gene Recognition Results

ROSETTA was then applied to our collection of 117 orthologous gene pairs. The program performed extremely well at identifying internal coding exons. Of internal coding exons, 94% were predicted perfectly at both ends and another 4% at one of the two ends. When one end is incorrectly predicted, the error typically involves only a few bases and typically is due to an alternative choice of splice site that more closely matches the expected pattern.

Only six of the internal coding exons (3%) were completely missed, and the reasons for the failures are instructive. (i) Three of these were in the galactose-1-phosphate uridyl transferase gene (Appendix C, Table C.1, gene 37). They resulted from the failure to

mouse intron 4, because 5'-splice site has a GC rather than the canonical GT. As a result, the gene is predicted to end at a downstream stop codon and a new gene is predicted to begin at an ATG codon upstream of exon 8. Exons 5, 6, and 7 are thus missed. (ii) Another exon is missed in the 21-hydroxylase gene (Appendix C, Table 1, gene 95), because the 5'-splice site is regarded as unlikely by our splice site detector: G-**GT**GCCTC in human, and T-**GT**TACCC in mouse. (iii) The two other internal coding exons that were missed are the instances in which two exons in one species correspond to a single exon in the other (in the Flt3 ligand (Appendix C, Table 1, gene 100) and lymphotoxin beta (Appendix C, Table 1, gene 85) genes, as noted above). The program's rules do not currently handle this special case.

The program was somewhat less accurate for initial and terminal coding exons. A total of 71% of such exons were correctly predicted at both ends. An additional 19% were correctly predicted at one end, with the incorrect end almost always being the initiation codon of an initial exon or the stop codon of a terminal exon. The errors typically involve predicting a splice site rather than the initiation or stop codon. In 17 cases, these splice sites are in fact annotated splice sites of the 5' and 3' UTRs.

A total of 2 initial and 9 terminal coding exons were completely missed. The initial coding exons were missed because they had length 3, consisting only of the ATG, which gave too weak a signal to detect. The terminal coding exons were missed because the coding exon was extremely short in one case (3 bp in human, 6 bp in mouse) or because the sequences were highly divergent between human and mouse.

Overall, the exon predictions were very accurate at the nucleotide level: 95% of nucleotides lying within coding exons were correctly predicted as such, and 97% of nucleotides predicted to lie within coding exons in fact did so. *ROSETTA* thus had 95% sensitivity and 97% specificity at the nucleotide level. *ROSETTA* predicted 26 coding exons that failed to overlap with any known exon.

We also compared our results with the performance of GENSCAN (Burge 1997). On our dataset, GENSCAN had similar nucleotide sensitivity (98%) but considerably lower nucleotide specificity (89%). Moreover, GENSCAN predicted 68 regions not overlapping any known coding exon, whereas *ROSETTA* predicted only 26 such instances.

3. Methods

3.1 Database Construction

A database of 1196 corresponding human/mouse mRNA pairs that had been previously compiled (Makolowski et al. 1996) was used to compile a database of 117 orthologous and annotated (with respect to gene structure) human and mouse genes. This was done by matching the human and mouse mRNA entries of the database to all human and mouse DNA entries in GenBank Release 109 (October 1998). Genes in the human or mouse that did not have a corresponding entry in the other organism were rejected. Entries were accepted only if they contained all the coding part of the gene, as well as the introns that lie between coding exons. Therefore, entries were accepted even if they did not constitute a complete gene, provided that they contained the coding part of the gene. In some cases, this caused us to accept genes without annotations or sequence for non-coding exons. Even though structural comparative information in non-coding regions could not be compiled for these entries, they were very useful for evaluating the quality of the *ROSETTA* coding region prediction method.

Our training set consisted of twelve pairs of human and mouse homologous genes, shown in Table 4.1. Training involved several steps. (1) Tuning parameters of *GLASS* in order to perfect the construction of global alignments of homologous genomic loci. (2) Choosing a PAM matrix for the protein alignments of pairs of potential exons. (3) Defining appropriate likelihood penalties for exons that were not preserved as is typical (for example exons whose length difference was not a multiple of 3).

3.2 Sequence Alignments and Comparative Analysis

When two corresponding regions in the human and mouse are not very similar, *GLASS* does not necessarily produce the exact map between the regions. For that reason, the corresponding introns, coding, and non-coding exon fragments in human and mouse sequences were further realigned using the standard dynamic programming alignment algorithm in order to compute more accurate local alignments for the purpose of compiling nucleotide similarity statistics.

Furthermore, the corresponding coding fragments were translated into protein and aligned using a PAM20 matrix obtained from the NCBI website (http://www.ncbi.nlm.nih.gov/) for the purpose of compiling protein similarity statistics.

Nucleotide similarity statistics in Appendix C, Table 1 for corresponding regions were computed using our *similarity count* SC(*,*) function. SC is a non-symmetric function that given two sequences s_1, s_2 (in our case in human and mouse, respectively) and an alignment between s_1 and s_2, returns the number of *valid matching positions* of s_1 into s_2. The number of valid matching positions is the number of positions j in s_1 that are mapped with a match to s_2 and such that either (1) j is the first or last position in s_1, or (2) $j - 1$ and $j + 1$ are not mapped to gaps in s_2. Thus, spurious matches in predominantly gapped regions do not add to the similarity count. This way the similarity count is not higher in the cases where the region s_2, in our case the mouse region, is much longer than s_1 and therefore s_1 can be aligned with many gaps and a large number of spurious matches. The similarity counts were divided by the lengths of the human regions.

Amino acid similarity statistics for corresponding coding regions were computed by counting the number of matching positions in the amino acid alignment of the regions, and dividing it by the length of the human exon.

Total sequence nucleotide similarity statistics were computed using the global alignment of the sequences derived by *GLASS*. A window of good alignment was defined to be a window of size 51 containing at least 20 matches. Any matches not contained inside a window of good alignment were discarded, and the number of remaining matches was divided by the length of the human locus.

3.3 Computational Prediction of Coding Regions

(i) *Masked Regions*. Before finding the optimal valid parse, the human and mouse sequences were preprocessed to mask repeats using the Repeatmasker program (available in http://ftp.genome.washington.edu/RM/RepeatMasker.html) and regions of weak alignment. A position was defined to be in a region of weak alignment if it was either (1) situated in a gap

of length at least 30, or (2) in the middle of a window of size 37 that contained fewer than 10 matches. Nucleotides in masked regions were disqualified from being predicted as coding.

(ii) *Splice Site Scores*. Splice site scores were computed using the directional rule modification to the GENSCAN splice site detector, as explained in (Pachter, 1999, Pachter et al. 1999). Donor splice site scores were multiplied by 0.5 and acceptor splice site scores were multiplied by 3.5. These values were obtained by requiring that the mean scores for donor and acceptor splice sites be equal in our training set (Pachter et al. 1999). Potential exons with a combined score of less than -10 for flanking splice sites were discarded in the dynamic programming algorithm (that is, they were given a score of $-\infty$).

(iii) *Coding Exon Length*. Corresponding potential coding exons with different lengths in the human and mouse sequence, were penalized as follows: initial and terminal exons with different length were given a penalty of –3 if lengths were equal mod 3, and –9 otherwise. For internal coding exons, the corresponding penalties were –9 and –27. These values were chosen heuristically, and were found to combine well with the other components of the scoring, most importantly the PAM20 matrix and the splice site scores. For instance, a PAM20 gap penalty is –19, while a PAM20 base substitution "penalty" ranges from +1 to –17, with typical values in the –8 range.

(iv) *Merging Forward and Reverse Complement Strand Parses*. Currently *ROSETTA* handles forward and reverse complement strands separately. Parses in the two strands are subsequently merged in a post-processing step. For each predicted exon e, a window extending 2000 positions in each direction from the endpoints of e, is used to count the forward and reverse complement coverage of the genomic region. That is, the number of predicted coding positions in each direction, included in the window, is calculated. If the direction of e is the direction with the highest count, e is accepted. Otherwise e is rejected. Future versions of *ROSETTA* may include a sophisticated genomic region model, where parses in both strands are simultaneously optimized.

For further details on the parameter selection for *ROSETTA* and *GLASS* we refer to our website (http://theory.lcs.mit.edu/crosspecies/).

3. Discussion

Analysis of large genomes is challenging because the important functional elements comprise only a small minority of the sequence: the problem is to extract signal from noise. Feature detectors that perform well enough in small genomes may become overwhelmed by large genomes and yield too many false positives.

A powerful solution is to first increase the signal-to-noise ratio by using evolutionary conservation among species. One can thereby focus attention on the portion of the sequence that is conserved (thereby decreasing noise) and search for features that are present in both species (thereby increasing the specificity of the signal).

Such strategies, of course, require that the elements to be found are indeed conserved by evolution. This certainly is the case for coding exons in human and mouse. Our study of the genomic structure of 117 orthologous gene pairs provides a quantitative description of the high degree of conservation in the number, length and sequence of coding exons.

The basic notion of using cross-species sequence comparison to identify important functional elements is well known, and has been used to study particular human and mouse regions (Hardison et al. 1997; Oeltjen et al. 1997; Jang et al. 1999). Gene recognition, however, does not emerge by simple inspection from the pattern of conservation: many non-genic elements are also well conserved, sometimes more so than genic elements. On average, the coding exons represented only a subset of the total well-aligned sequence.

We sought to develop an automatic method for gene recognition given cross-species sequence. The approach involves aligning the genomic sequences and then parsing the sequences to find a gene model in which the proposed exons are supported by features (splice sites, codon usage, etc.) present in both species. Alignment is performed with the *GLASS* program and gene recognition with the *ROSETTA* program. Both programs are available for use on a public web server (http://theory.lcs.mit.edu/crosspecies/).

The resulting program identifies the location of coding exons with high specificity and sensitivity. The vast majority of coding exons are identified perfectly. The overall results were robust across genes, including instances such as the tumor necrosis factor beta gene in which

the first intron shows higher conservation than the flanking exons. The remaining errors largely result from highly unusual features — such as rare splice signals or fused exons.

ROSETTA represents only a first attempt at systematically using cross-species information for gene recognition. It should be possible to refine the program by incorporating feature detectors used in single-species gene recognition programs (such as those for promoters, poly-adenylation sites, etc, as well as more sophisticated statistical tests), by refining the way in which the existing detectors are combined and by incorporating rules to detect special cases (such as fused exons or non-canonical splice sites). The program is designed to recognize a single optimal gene model; a further challenge would be to recognize conserved patterns of alternative splicing by exploiting backtracking features of dynamic programming.

Our list of 117 orthologous pairs studied is necessarily biased toward genes with smaller genomic loci, owing to the fact that genomic sequence from such loci is over represented in current databases. Such a bias towards shorter genes could potentially enhance *ROSETTA*'s performance because of a higher signal-to-noise ratio in such genes. Nonetheless, the list contains many larger genomic loci and *ROSETTA* performs as well on the larger loci as on the smaller ones. It will be important to continue to test the program on large loci, as they become available.

An interesting question is whether the mouse is a suitable organism to select for exon prediction in human genes. Organisms whose sequence has not drifted sufficiently far from the human will not increase the signal-to-noise ratio sufficiently, while organisms that are too distant may make it difficult to recognize important signals. Interestingly, *ROSETTA* produced roughly equal amounts of over-prediction and under-prediction, which may suggest that the human and mouse are at a felicitous distance for the purpose of coding exon prediction.

With the explosion in the sequencing of the human and mouse genomes, cross-species sequence comparison should become an increasingly important technique for extracting information from the mammalian genome. We demonstrate here a systematic technique for extracting the vast majority of the information about coding exons. The next challenge will be to create similar systematic techniques to extract information about non-coding exons, promoters, regulatory elements and other important functional features of the genome.

CONCLUSION

In this thesis we discussed the steps involved in the planning and execution of a large sequencing project, and the subsequent steps of first-order annotation and global sequence comparison of the finished genomic sequence.

We presented contributions in all the above steps of a genome project:

In the steps of planning and sequencing a genome, we presented contributions in the major alternative methods for whole-genome sequencing. (1) We first studied the Hypergraph Superstring problem, a mathematical problem motivated by physical mapping, the main clone-by-clone approach for sequencing a genome. Our complexity results on the *MIN-Hypergraph-Superstring* and on the *MIN-Sperner-Decomposition* problems, and our approximation algorithms, may be interesting from a theoretical computing perspective irrespective of their application to physical mapping. (2) We studied an important method for sequencing a large genome: walking with BAC-ends. Our statistical analysis and simulations provide predictions of performance, and suggest optimal tradeoffs between sizes of clone libraries used, number of walking phases, and resulting sequencing quality and efficiency. (3) We studied the computational problem of assembling shotgun sequencing data. Our prototype system *ARACHNE* shows considerable success in obtaining a rough assembly of the data using computational resources that are within the means of current high-end computers. *ARACHNE* may be a starting point for building an elaborate assembly system to be used by academic sequencing efforts to assemble a mammalian genome such as the mouse genome.

We then turned to the important problem of studying the finished genomic sequence, using computational techniques. (1) We first presented *GLASS*, a system for obtaining the global alignment of pairs of long homologous DNA stretches. We demonstrated that the system succeeds extremely well in aligning human and mouse homologous genomic loci that are similar around 60-95% in the short exon regions, but almost unrelated in the much longer intron regions between the exons. *GLASS* should also be appropriate for obtaining the global alignment of homologous pairs of sequences that are more similar than human and mouse orthologous loci. It may prove useful in whole-genome self-similarity studies and we may use it for this purpose on the human genome draft that will soon be available. However, *GLASS*

may be inappropriate to use for aligning more distant homologs, as it relies in the presence of several matching *k-mers* between the two input sequences. (2) We used *GLASS* to perform a study of 117 homologous human and mouse genomic loci. We provided structural similarity statistics for human and mouse genes, and suggested the use of them in genomic annotation using sequence comparisons. (3) Finally, we built *ROSETTA*, the first cross-species comparison-based gene recognition system. We demonstrated that a very simple implementation of such a system compares favorably to sophisticated gene recognition programs that use information from one species alone.

We would like to conclude with some remarks on future work.

We believe that the main interesting open questions on Chapter 1 are theoretical: to completely determine the approximation complexity of the *MIN-Hypergraph-Superstring* and *MIN-Sperner-Decomposition* problems. We have shown that the problems are at least *MAXSNP*-hard, but it is still open to either give constant approximation algorithms, or prove stronger inapproximability results.

We have barely touched the expanding field of global sequence comparisons, and comparison-based genome annotation. There are several possible avenues of future work. Here we will mention only a few: (1) Refinement of *GLASS* and application to many different pairs of organisms, and also to self-similarity studies of genomes. Such studies could reveal important evolutionary facts, and/or evidence for large-scale genome duplications and rearrangements. (2) Expansion of the gene recognition methods into recognition of other evolutionarily preserved signals such as promoters and regulatory elements. (3) Development and application of similar annotation methods for different pairs of organisms (4) Development of similar methods for the simultaneous comparison and annotation of three, or more organisms.

Several parts of this thesis are motivated by the search for better methods for whole genome sequencing. We would like to conclude with a description of what we believe may be a powerful method of sequencing and assembling large genomes in the future. In particular we believe that while whole genome shotgun assembly may be possible for a large genome such as

a mammalian genome, the fragment assembly problem may become much easier, and result in better sequence fidelity, if combined with clone-by-clone data. The academic sequencing community may believe the same, since the mouse genome will be obtained by a combination of whole genome shotgun and clone-by-clone sequencing (Science, vol. 287, p. 1179, 2000). However what has not been systematically addressed yet is the exact method with which the data will be used to assemble the mouse sequence. Hence neither the optimal ratio of shotgun vs. clone-by-clone sequencing, nor the optimal combination of insert sizes for shotgun (to get the forward/reverse linking information), are known.

We would like to sketch here a plausible procedure by which a large genome could be sequenced and assembled using a specific combination of Whole Genome Shotgun and Clone-by-Clone (*WGSCC*) data. (Appendix D contains a more detailed description.)

First we would like to describe a plausible mix of data that could be obtained, aiming at a total amount of sequencing equal to the current standards, of 10x coverage of the genome with reads:

1. The majority of this sequence, say Kx where K is a constant < 10, is in the form of whole genome shotgun data, consisting of mostly short plasmid paired reads while possibly using some cosmid paired reads, or unpaired reads.
2. A random collection of BAC clones covers the genome to Lx, each clone being sequenced to a small Mx coverage.
3. A deep library of end-sequenced BAC clones is formed. If the library is 20-deep, if sequencing a read off the end of a BAC clone is twice as expensive as sequencing a regular plasmid or cosmid read, and if BAC clones are on average 150kb-long, this costs 0.25x of whole genome sequencing.

Typical values for K, L, and M would be 7, 2, and 0.5, respectively. Then, a majority of the genome would be covered with BAC clones that are obtained in step 2 above, and are each sequenced to 0.5x. This would correspond to having roughly 150 reads per clone, or one read per 1,000 bases. Assume for the moment that the full sequence of most of these clones can be reliably obtained using these local reads, in combination with the 7x coverage of the whole

genome with shotgun reads.[1] Then, the genome would be covered with *islands* of fully assembled sequence, separated with *oceans* of unassembled genome. The library of end-sequenced BACs could then be employed to walk from these islands similarly to the way described in Chapter 2. In this case though, it would not be as important to obtain a minimal overlapping clone from the library. The library clones selected for walking would only be sequenced to 0.5x depth resulting in virtually no wastage due to overlaps with already assembled islands. The whole scheme as described above involves performing a total sequencing of around 8.5x. This is lower than the "allowable" 10x total sequencing, and therefore there is considerable slack that can be devoted to sequence the most "difficult" regions of DNA. Specific regions can receive special attention because they are "captured" in BAC clones that have been end-sequenced or lightly shotgun sequenced. For instance, the hardest 5% of the genome consisting of repeats that are hard to resolve and assemble with 99.99% accuracy with whole genome shotgun could be sequenced clone-by-clone to depth ~5x. The total additional sequencing this step would involve is only 0.25x, well within the allowable slack. Then, the 5x sequencing of these clones together with the 7x whole genome shotgun should enable to yield the best possible quality of assembly in these regions, similar to 10x clone-by-clone sequencing. One can see that the *WGSCC* approach allows for considerable control over the quality of the final sequence, exactly because a clone-by-clone map of the genome is obtained in parallel with the consensus sequence.

We plan to develop the prototype system *ARACHNE* into a finished software package that is capable of performing fragment assembly in the presence, or absence, of clone-by-clone data as described above. That would first of all involve implementing an algorithm as the one described in Appendix D, which is capable of utilizing WGSCC data. It would also involve several important refinements of our existing software: using *PHRED* scores in pairwise read alignments; producing finished sequence instead of a rough assembly of reads into supercontigs; refinement and speedup of the algorithms; development and parallelization in a suitable platform; and possibly several other crucial steps.

---

[1] In fact assembling the sequence of such a clone ought to be considerably easier using the 150 reads from the clone, than it would be if the only data available were 7.5x shotgun sequence of the whole genome.

Appendix A

The Genetic Code

The ribosome converts mRNA into protein according to the genetic code. Three nucleotides are converted to one amino acid. Protein synthesis starts from an occurrence of the initiation codon (ATG) and ends with the first occurrence of a termination codon (TAA, TAG, or TGA) in the translating frame.

	T	C	A	G
T	TTT Phe (F) TTC * TTA Leu (L) TTG *	TCT Ser (S) TCC * TCA * TCG *	TAT Tyr (Y) TAC * TAA *Stop* TAG *Stop*	TGT Cys (C) TGC * TGA *Stop* TGG Trp (W)
C	CTT Leu (L) CTC * CTA * CTG *	CCT Pro (P) CCC * CCA * CCG *	CAT His (H) CAC * CAA Gln (Q) CAG *	CGT Arg (R) CGC * CGA * CGG *
A	ATT Ile (I) ATC * ATA * ATG Met (M)	ACT Thr (T) ACC * ACA * ACG *	AAT Asn (N) AAC * AAA Lys (K) AAG *	AGT Ser (S) AGC * AGA Arg (R) AGG *
G	GTT Val (V) GTC * GTA * GTG *	GCT Ala (A) GCC * GCA * GCG *	GAT Asp (D) GAC * GAA Glu (E) GAG *	GGT Gly (G) GGC * GGA * GGG *

Table A.1. The Genetic Code.

Appendix B

Performance of *ARACHNE*

In the tables below we provide the results on the performance of *ARACHNE* on all our test runs. Please refer to the main text (Chapter 3) for the meanings of Coverage, CERR, RERR, RERR2, GAP, RGAP, and ROUT. #SCON is the number of supercontigs above SC Length, and #CON is total the number of contigs in these long supercontigs.

Human, 11x									
Sc Length	#SCON	#CON	Coverage	CERR	RERR	RERR2	GAP	RGAP	ROUT
1000000	11	1869	51.6575	0.267523	0.121643	0.104435	0.951052	0.009267	2.07779
250000	30	3117	81.8569	0.737889	0.296759	0.182583	1.44267	0.012559	2.03651
100000	46	3506	90.1023	1.79692	0.325442	0.190764	1.66472	0.022858	2.02364
25000	82	3879	94.6051	2.83578	0.776246	0.448309	2.44104	0.117326	2.03509
10000	108	4017	95.8134	2.96241	0.91559	0.552387	2.71454	0.203166	2.04511
5000	141	4101	96.2259	3.02365	0.986972	0.611251	2.95164	0.374266	2.05136
2000	211	4220	96.5669	3.34123	1.11745	0.693346	3.25165	0.670593	2.0638

Table 1. Performance of *ARACHNE* on human chromosome 22, 11x coverage.

Human, 9x									
SC Length	#SCON	#CON	Coverage	CERR	RERR	RERR2	GAP	RGAP	ROUT
1000000	9	1859	51.0186	0.053792	0.03538	0.029433	0.993216	0.013933	1.80048
250000	28	2989	81.1249	0.16728	0.097715	0.084798	1.35776	0.010587	1.78148
100000	44	3323	88.9773	0.150466	0.106079	0.094278	1.5358	0.019265	1.78906
25000	78	3684	93.9008	1.00434	0.510291	0.255239	2.21059	0.096817	1.7857
10000	102	3812	94.9333	1.60021	0.790593	0.428751	2.49243	0.186535	1.78851
5000	134	3890	95.3919	1.79949	0.889088	0.495323	2.69901	0.351437	1.79182
2000	193	3990	95.775	2.10526	1.01624	0.569789	2.9524	0.59603	1.79655

Table 2. Performance of *ARACHNE* on human chromosome 22, 9x coverage.

Human, 7x									
SC Length	#SCON	#CON	Coverage	CERR	RERR	RERR2	GAP	RGAP	ROUT
1000000	8	1971	46.1732	0	0	0	1.3558	0.014499	1.46109
250000	30	3452	80.1969	0.057937	0.037863	0.024349	1.64758	0.020045	1.48063
100000	46	3824	87.4488	0.156904	0.11998	0.099143	1.93451	0.035474	1.46937
25000	77	4125	92.3115	0.606061	0.433768	0.273264	2.28699	0.068087	1.47533
10000	109	4268	93.9096	0.937207	0.652197	0.424052	2.57763	0.165069	1.4793
5000	150	4367	94.5611	1.14495	0.809359	0.556805	2.76095	0.301605	1.47879
2000	223	4461	95.0031	1.18807	0.912839	0.638131	2.87392	0.419139	1.48756

Table 3. Performance of *ARACHNE* on human chromosome 22, 7x coverage.

Human, 5x									
SC Length	#SCON	#CON	Coverage	CERR	RERR	RERR2	GAP	RGAP	ROUT
250000	16	1003	16.7055	0	0.012878	0.012878	5.17857	0.236511	1.01796
100000	109	3640	59.1112	0.192308	0.175004	0.070327	6.07925	0.346571	1.04065
25000	277	5489	85.6575	0.564766	0.513473	0.245193	6.3743	0.368636	1.05154
10000	373	5886	90.1071	0.696568	0.725086	0.391209	6.69131	0.474651	1.06078
5000	462	6081	91.4638	0.74001	0.844548	0.512033	6.87023	0.68757	1.06279
2000	671	6318	92.6469	0.79139	1.02212	0.64133	6.84246	0.769325	1.07202

Table 4. Performance of *ARACHNE* on human chromosome 22, 5x coverage.

Human, 3x									
SC Length	#SCON	#CON	Coverage	CERR	RERR	RERR2	GAP	RGAP	ROUT
25000	50	329	4.72639	0	0.366415	0.366415	18.4532	3.16608	0.582848
10000	496	1895	24.4016	0.686016	1.24275	0.892227	19.3543	5.82302	0.64801
5000	1396	3801	42.1679	0.552486	1.41661	1.11506	15.935	6.55698	0.714661
2000	3481	5979	58.3367	0.401405	1.53955	1.28717	11.769	5.04107	0.699569

Table 5. Performance of *ARACHNE* on human chromosome 22, 3x coverage.

C.elegans, 11x									
SC Length	#SCON	#CON	Coverage	CERR	RERR	RERR2	GAP	RGAP	ROUT
1000000	3	484	37.8117	0.413223	0.593201	0.482119	2.11728	0.00389	1.83739
250000	15	1268	79.2325	0.630915	0.643322	0.520089	2.74224	0.015964	1.95141
100000	29	1702	96.1433	1.64512	1.21876	0.812193	3.46796	0.014012	1.99304
25000	33	1759	97.841	1.59181	1.25085	0.851715	3.48182	0.013765	1.99171
10000	37	1781	98.2516	2.02134	1.38385	0.923465	3.70871	0.138385	1.99164
5000	42	1795	98.5143	2.22841	1.48353	1.00047	3.80176	0.203332	1.99229
2000	49	1810	98.5938	2.26519	1.53518	1.05018	3.85746	0.26253	1.99449

Table 6. Performance of *ARACHNE* on C. elegans chromosome 1, 11x coverage.

C.elegans, 9x									
SC Length	#SCON	#CON	Coverage	CERR	RERR	RERR2	GAP	RGAP	ROUT
1000000	2	399	36.0866	8.5213	0.59539	0.576362	2.04595	0.008161	1.44865
250000	14	1307	80.9604	3.36649	0.709746	0.584082	3.19038	0.015718	1.46863
100000	26	1639	94.8184	3.17267	1.07257	0.837145	3.76298	0.019756	1.51286
25000	33	1732	97.499	3.52194	1.18568	0.914727	4.02217	0.07801	1.50727
10000	37	1750	97.7769	3.71429	1.28042	0.990229	4.14704	0.147317	1.50858
5000	47	1776	98.1206	3.71622	1.36847	1.08131	4.2263	0.215473	1.51782
2000	57	1791	98.1938	3.74093	1.42934	1.13924	4.27348	0.266396	1.51923

Table 7. Performance of *ARACHNE* on C. elegans chromosome 1, 9x coverage.

H. influenzae, 11x									
SC Length	#SCON	#CON	Coverage	CERR	RERR	RERR2	GAP	RGAP	ROUT
1000000	1	67	86.4665	0	0.357992	0.357992	2.12391	0.010928	1.36494
100000	2	78	99.711	0	0.311869	0.311869	2.1178	0.009477	1.35722
5000	3	80	99.711	0	1.22267	1.22267	2.15926	0.031893	1.3644

Table 8. Performance of *ARACHNE* on the H. influenzae genome, 11x coverage.

H.influenzae, 9x									
SC Length	#SCON	#CON	Coverage	CERR	RERR	RERR2	GAP	RGAP	ROUT
1000000	1	78	86.3928	1.28205	0.391347	0.330261	2.04927	0.00076	1.14271
100000	2	91	99.6368	1.0989	0.339633	0.28656	2.07378	0.000659	1.1471
5000	3	93	99.9196	1.07527	1.47565	1.42485	2.07958	0.002189	1.16129

Table 9. Performance of *ARACHNE* on the H. influenzae genome, 9x coverage.

H.influenzae, 7x									
SC Length	#SCON	#CON	Coverage	CERR	RERR	RERR2	GAP	RGAP	ROUT
1000000	1	76	86.3722	1.31579	0.392627	0.331343	2.32361	0.018239	1.13575
100000	2	88	99.6019	1.13636	0.340737	0.287492	2.3432	0.021527	1.10452
5000	3	91	99.912	1.0989	1.48024	1.42928	2.33619	0.021459	1.14056

Table 10. Performance of *ARACHNE* on the H. influenzae genome, 7x coverage.

H. influenzae, 5x									
SC Length	#SCON	#CON	Coverage	CERR	RERR	RERR2	GAP	RGAP	ROUT
250000	2	130	51.4554	0	0.042932	0.042932	4.70981	0.257302	0.777423
100000	5	203	78.0981	0	0.057954	0.057954	5.26323	0.337643	0.647762
25000	10	258	97.1	0.387597	0.431085	0.431135	5.19668	0.32288	0.688458
5000	11	260	97.1	0.384615	1.67383	1.67402	5.18229	0.321847	0.700137
2000	19	268	97.4689	0.373134	1.75329	1.75349	5.12246	0.318131	0.704265

Table 11. Performance of *ARACHNE* on the H. influenzae genome, 5x coverage.

A.fulgidus, 7x									
SC Length	#SCON	#CON	Coverage	CERR	RERR	RERR2	GAP	RGAP	ROUT
1000000	1	215	99.9589	0	0.023263	0.023263	0.909781	0.01539	1.28596
5000	2	216	99.9589	0	0.069652	0.069652	0.907585	0.015353	1.28348

Table 12. Performance of *ARACHNE* on the A. fulgidus genome, 7x coverage.

Below we provide the figures of statistics of the assembly quality, for each of the above tests. These are in exactly the same format as the corresponding figures in Chapter 3. The figures in Chapter 3 are duplicated here in order to provide a complete reference to all our tests.

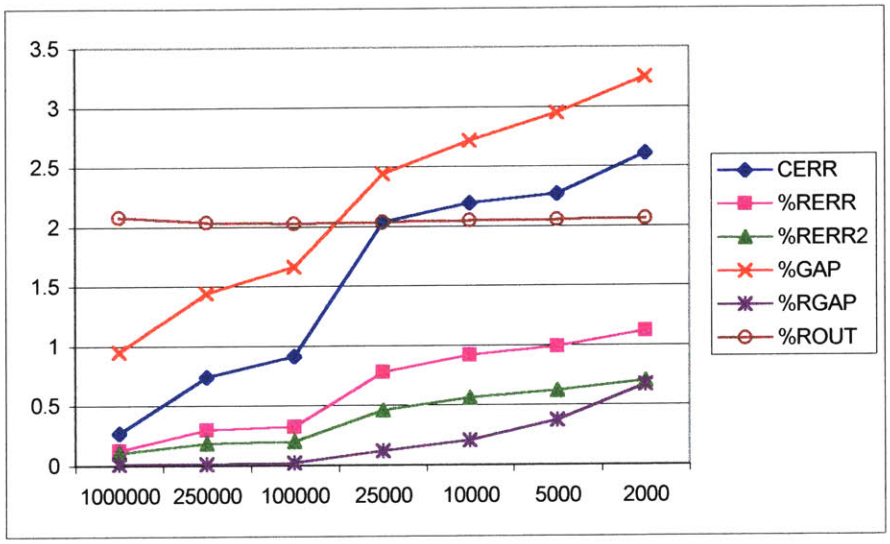

Figure 1. Quality of shotgun assembly on human chromosome 22, 11x coverage with shotgun reads.

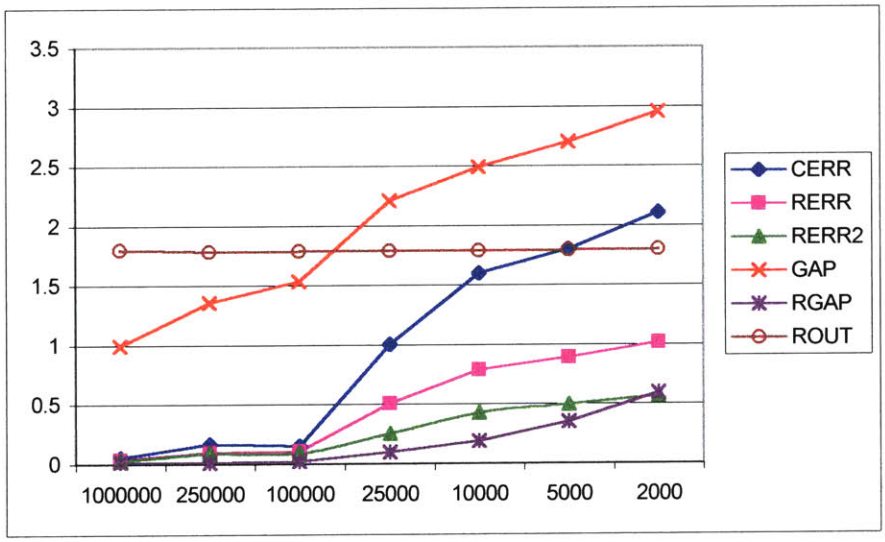

Figure 2. Quality of shotgun assembly on human chromosome 22, 9x coverage with shotgun reads.

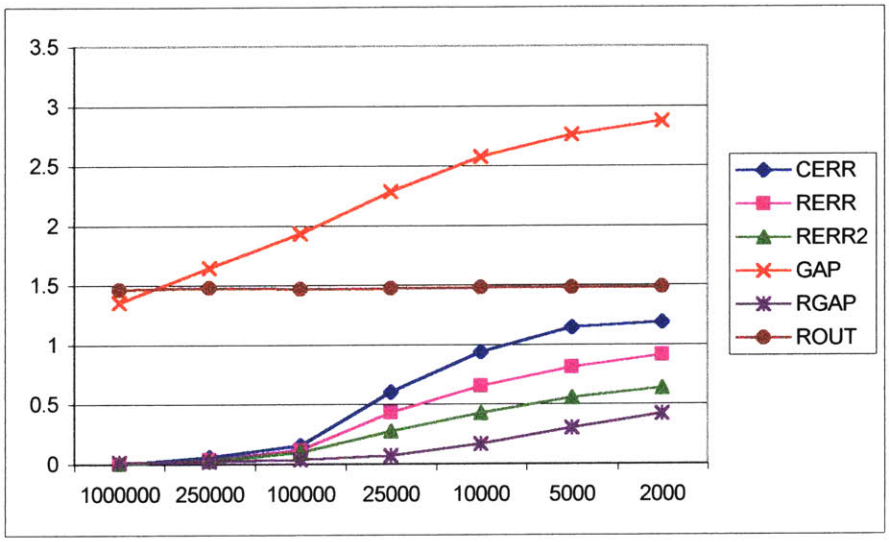

Figure 3. Quality of shotgun assembly on human chromosome 22, 7x coverage with shotgun reads.

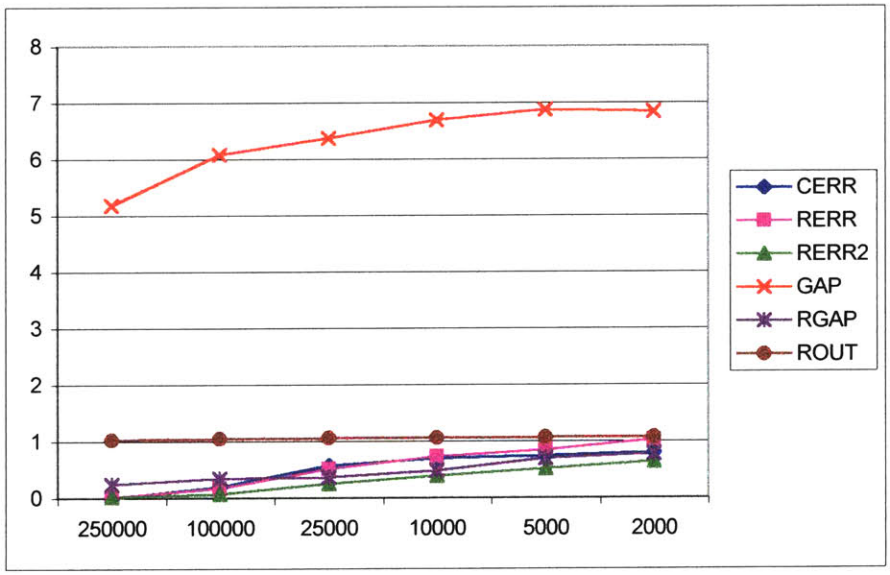

Figure 4. Quality of shotgun assembly on human chromosome 22, 5x coverage with shotgun reads.

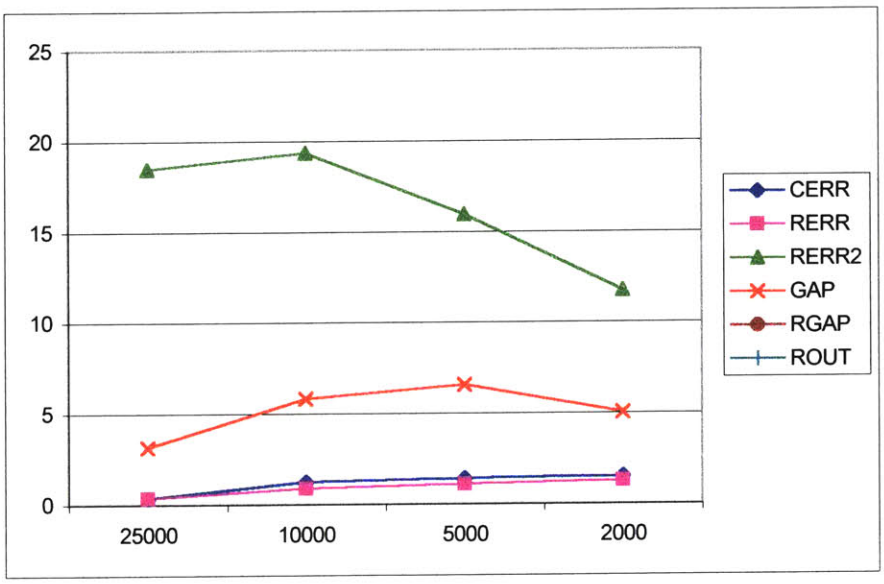

Figure 5. Quality of shotgun assembly on human chromosome 22, 3x coverage with shotgun reads.

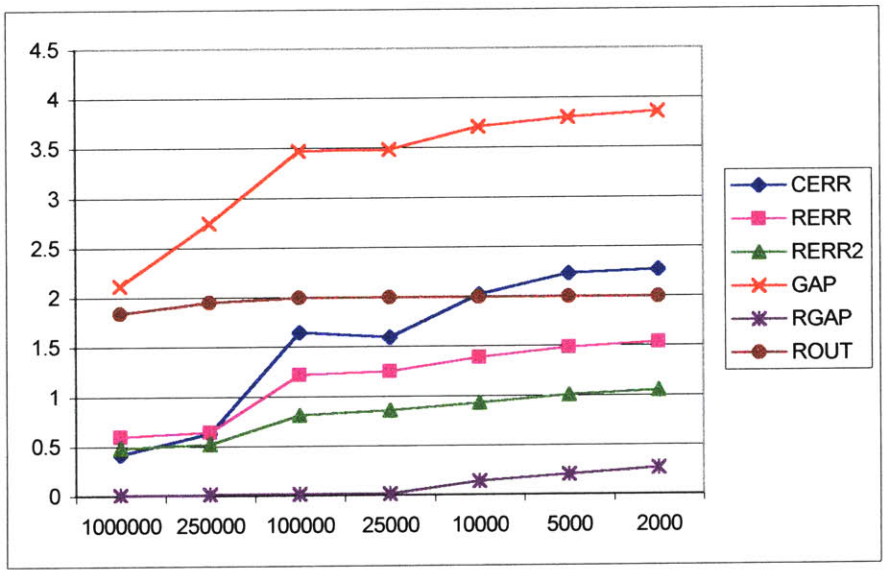

Figure 6. Quality of shotgun assembly on C. elegans chromosome I, 11x coverage with shotgun reads.

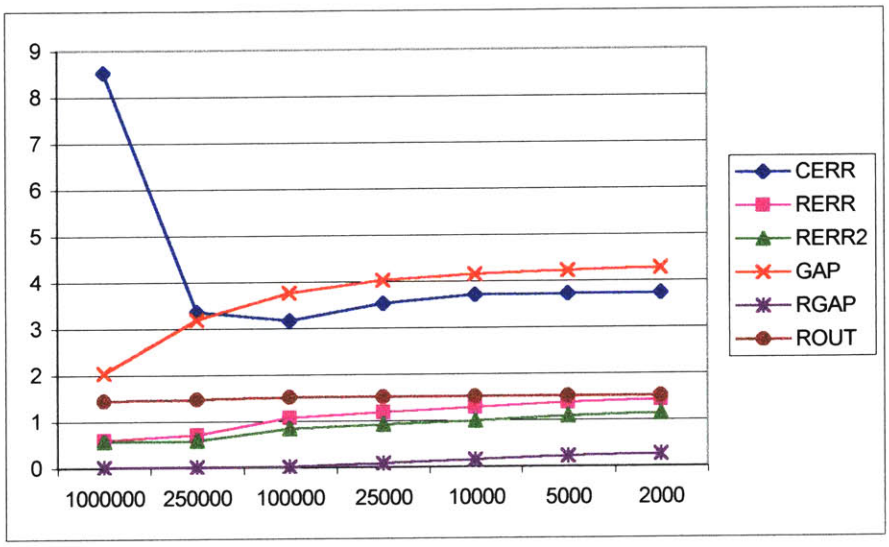

Figure 7. Quality of shotgun assembly on C. elegans chromosome I, 9x coverage with shotgun reads.

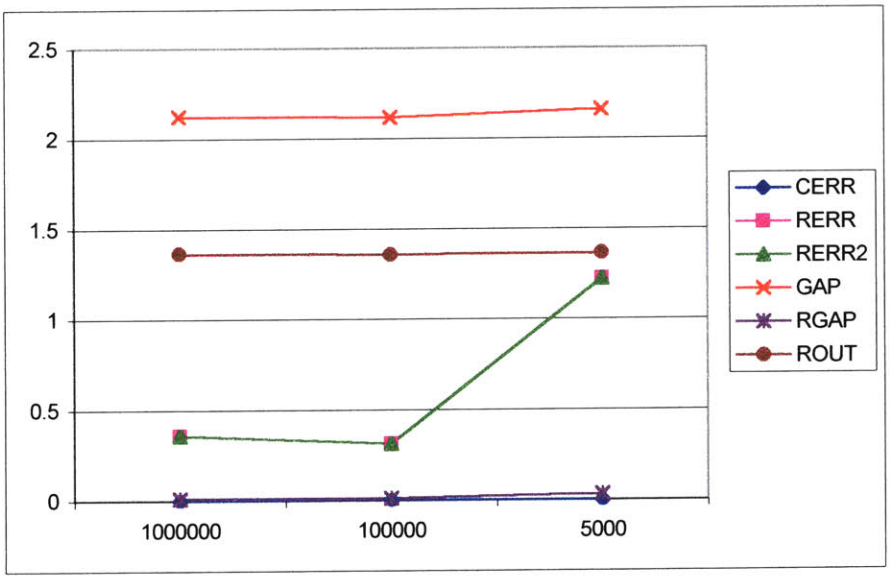

Figure 8. Quality of shotgun assembly on the H. influenzae genome, 11x coverage with shotgun reads.

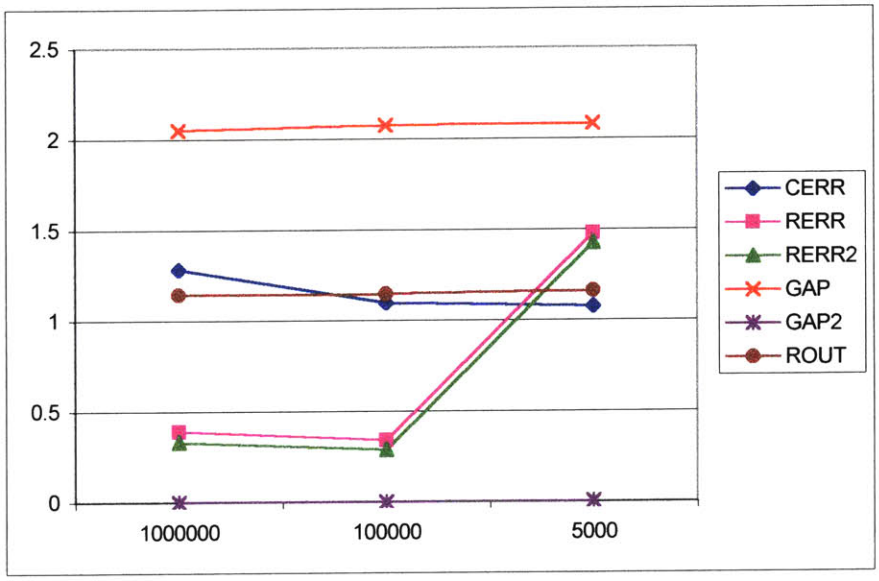

Figure 9. Quality of shotgun assembly on the H. influenzae genome, 9x coverage with shotgun reads.

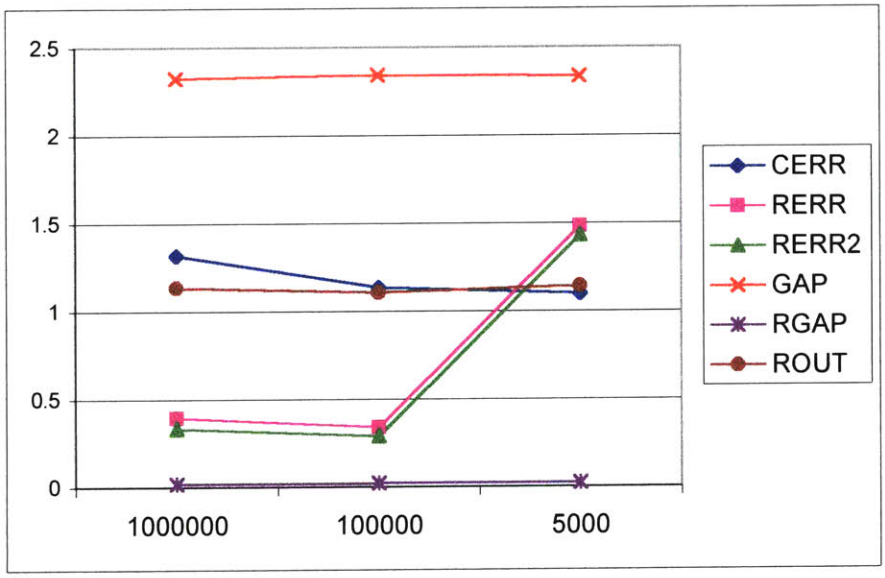

Figure 10. Quality of shotgun assembly on the H. influenzae genome, 7x coverage with shotgun reads.

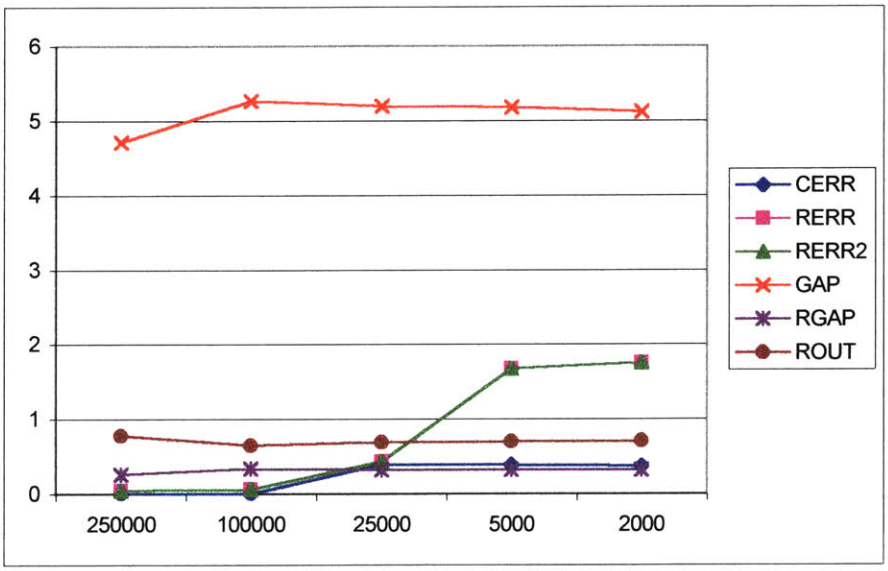

Figure 11. Quality of shotgun assembly on the H. influenzae genome, 5x coverage with shotgun reads.

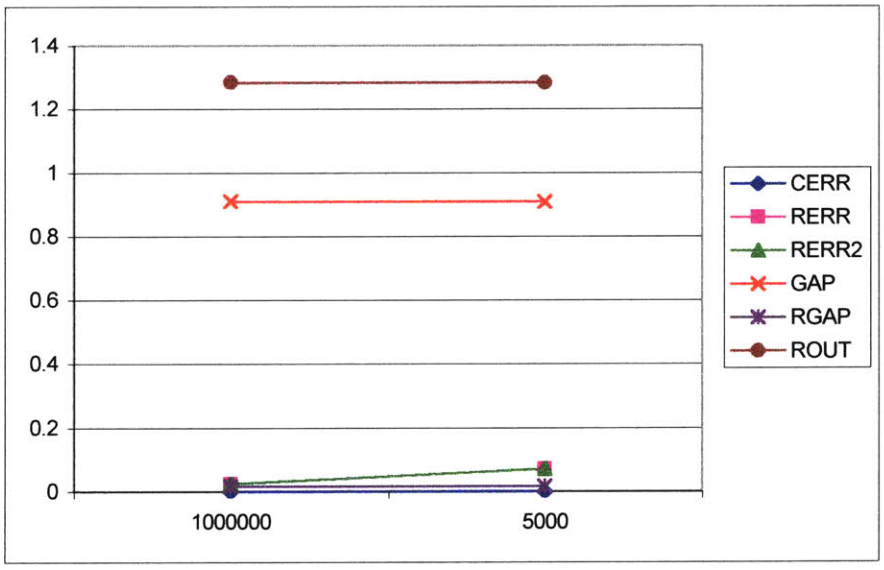

Figure 12. Quality of shotgun assembly on the A. fulgidus genome, 7x coverage with shotgun reads.

Appendix C

Comparative Analysis of Human and Mouse Loci

In the table below (Table C.1) we report a structural comparison of 117 orthologous human and mouse genomic loci. We also report the exon prediction performance of ROSETTA on each of these loci.

Each entry in the table, numbered 1-117, is a pair of orthologous loci. In the first column, the Genbank LOCUS of the human entry, followed by the Genbank LOCUS of the mouse entry, followed by short descriptions of the genes, are given. The following columns have the following meanings, depending on the rows:

1. First row corresponds to the human entry.
2. Second row corresponds to the mouse entry.
3. Third row corresponds to nucleotide sequence similarity.
4. Fourth row, when applicable, corresponds to amino acid similarity.
5. Fifth row, when applicable, corresponds to ROSETTA predictions.

Thus the columns have the following meaning:

1. Third column, colored dark, corresponds to the total size for human and mouse, and the total sequence similarity using the GLASS alignment.
2. Fourth column corresponds to the sizes, and nucleotide similarity of the 5'-UTRs.
3. Fifth column corresponds to the sizes, nucleotide, and protein similarity of the translated regions.
4. Sixth column corresponds to the sizes, and nucleotide similarity of the 3'-UTRs.
5. Seventh column corresponds to the sizes, and nucleotide similarity of the introns.
6. The rest of the columns correspond to the sizes, nucleotide similarity, and protein similarity plus ROSETTA predictions whenever applicable. The color shading the regions indicates the type of the regions: coding exons (white), noncoding exons (light gray), and introns (medium dark gray).

The ROSETTA predictions are indicated as follows:

++: Coding exon predicted correct on both ends.

+-: Coding exon predicted correct only on the 5'-end.

-+: Coding exon predicted correct only on the 3'-end.

--: Coding exon was not missed totally, but both 3'- and 5'- boundaries were wrongly predicted.

X: Coding exon was missed altogether.

Structurally unusual cases, such as when two coding exons in human correspond to one in mouse, can readily be seen in the table. For instance entry 30 has such a situation. Coding exons 5 and 6 in human can be seen to correspond to coding exon 5 in mouse.

		Total	5' NC	C.ex.	3'NC	Intron	Alignments of Regions																					
1	HSCKIIBE	5917	340	648	140	3065	329	611	11	72	966	103	569	116	446	76	146	190	327	91	140							
	MMGMCK2B	7874	332	648	140	4129	321	593	11	72	588	103	1180	116	387	76	110	190	1271	91	140							
	Casein kinase II subunit beta gene	57.51	66.07	93.06	77.14	40.89	66.15	56.31	63.64	94.44	31.79	95.15	36.13	89.66	50.9	92.11	54.69	93.16	27.41	94.51	77.14							
				98.61						100		99.03		98.28		98.68		97.89		98.9								
										-+		++		++		++		++		++								
2	HUMSAACT	3778	103	1134	253	1345	91	869	12	129	106	325	127	162	79	192	86	182	78	144	253							
	MUSACASA	4007	70	1134	242	1498	58	961	12	129	98	325	140	162	93	192	132	182	74	144	242							
	Skeletal alpha-actin gene	59.53	41.34	90.3	53.7	31.75	33.6	26.12	100	89.15	44.12	88.92	36.7	89.51	45.35	90.1	51.38	92.31	34.21	93.06	53.7							
				99.21						100		99.69		98.15		98.44		98.9		100								
										-+		++		++		++		++		++								
3	HSH4EHIS	859	NA	312	NA	NA	312																					
	MMHIS412	637	NA	312	NA	NA	312																					
	Histone H4 gene	57.61	NA	89.42	NA	NA	89.42																					
				100			100																					
							++																					
4	HSU12202	4942	36	393	81	3921	36	3	1443	66	93	210	1380	111	405	3	19	600	62	-	-							
	MMMRPS24	5499	46	396	200	4587	46	3	990	66	91	210	1265	111	558	6	12	1358	20	325	84							
	Ribosomal protein S24 gene	34.42	41.46	88.55	4.539	16.9829	41.46	100	23.76	84.85	43.48	89.05	16.64	90.99	25.75	66.7	19.35	29.72	0	0	0							
				100				100		100		100		100		100												
								X		++		++		++		X												
5	HUMHIS4	1098	NA	312	NA	NA	312																					
	MUSHIST4	968	NA	312	NA	NA	312																					
	Histone H4 gene	48.86	NA	87.18	NA	NA	87.18																					
				100			100																					
							-+																					
6	HSHISH3	698	NA	411	NA	NA	411																					
	MMHIST31	592	NA	411	NA	NA	411																					
	Histone H3 gene	72.13	NA	85.89	NA	NA	85.89																					
				100			100																					
							++																					
7	HSHSC70	5408	83	1941	NA	2380	78	730	5	205	322	206	324	153	87	556	211	203	228	199	147	233	331	186	NA			
	MMU73744	4270	65	1941	85	1840	60	566	5	205	308	206	115	153	84	556	212	203	222	199	95	233	238	186	85			
	Hsc70 gene for heat shock cognate protein	61.62	52.47	88.97	NA	30.23	50.7	26.1	80	86.34	22.86	89.32	13.67	90.2	47.95	89.57	43.5	92.12	52.44	84.92	14.05	89.7	41.48	88.71	NA			
				99.23						99.51		99.03		100		99.82		97.54		99.5		97.85		100				
										-+		++		++		++		++		++		++		++				
8	HUMNOCT	4878	NA	1332	NA	NA	1332																					
	MUSPOUDOMB	3864	NA	1338	NA	NA	1338																					
	POU domain transcription factor	71.58	NA	93.84	NA	NA	93.63																					
				99.1			99.1																					
							++																					
9	HUMTROC	4567	27	486	173	2244	27	24	1466	31	230	147	248	115	216	137	84	32	173									
	MUSCTNC	4194	44	486	173	2687	44	24	1418	31	226	147	354	115	602	137	87	32	173									
	Slow twitch skeletal nuscle cardiac troponin	54.41	62	89.92	59	48.90	62	91.67	54.09	83.87	47.37	91.16	32.23	92.17	35.7	86.86	45.61	93.75	59									
				96.91				100		96.77		97.96		96.52		96.35		93.75										
								++		++		++		++		++		++										
10	HSINT1G	4522	NA	1113	NA	1877	104	713	254	702	266	462	489															
	MUSINT1A	5607	NA	1113	NA	1599	104	569	254	573	266	457	489															
	Int-1 mammary oncogene	65.33	NA	91.11	NA	46.21	90.38	41.97	88.98	49.57	91.35	47.66	92.23															
				98.38			89.42		98.03		99.25		100															
							-+		++		++		++															

: Coding Exons

: Noncoding Exons

: Introns

11	HUMSRI1A MUSSRI1A Somatostatin receptor isoform 1 gene	1634 1265 87.59	NA NA NA	1176 1176 89.12 98.72	NA NA NA	NA NA NA	1176 1176 89.12 98.72 ++																				
12	HSMIMAR MUSACHRM1 M1 gene for muscarinic acetylcholine	2100 1574 86.59	NA NA NA	1383 1383 90.31 98.05	NA NA NA	NA NA NA	1383 1383 90.31 98.05 -+																				
13	HSFAU1 MUSFAUA Fan 1 gene	2016 2850 63.19	105 61 51.81	402 402 90.8 97.01	NA 450 NA	998 1066 42.69	97 53 49.33	269 241 51.76	8 8 75	75 75 94.67 96 -+	94 102 37.76	145 145 88.28 95.17 ++	461 572 34.08	56 56 89.29 96.43 ++	174 151 54.15	126 126 92.06 100 ++	NA 450 NA										
14	HUMCRYABA MUSALPBCRY Alpha-B-crystallin gene	4206 4181 62.86	50 45 73.7	528 528 90.72 97.73	NA 138 NA	2431 2705 47.65	50 45 73.7	201 201 92.04 95.52 -+	1074 1048 53.1	123 123 83.74 100 ++	1357 1657 43.33	204 204 93.63 98.53 ++	NA 138 NA														
15	HSENO3 MMENO3G ENO3 gene for muscle specific enolase	7194 5472 59.5	108 66 51.72	1305 1305 89.04 96.09	80 79 71.7	4512 3856 38.91	106 64 51.76	603 602 48.8	2 2 50	85 85 94.12 98.82 -+	876 206 20.15	96 96 88.54 93.75 ++	160 128 47.22	59 59 86.44 81.36 ++	161 147 56.49	70 70 91.43 98.57 ++	369 360 43.62	134 134 85.82 96.27 ++	1231 1499 32.97	223 223 94.17 99.55 ++	110 114 65.18	198 198 85.35 93.94 ++	337 194 41.43	202 202 89.11 96.53 ++	428 372 51.25	109 109 88.99 99.08 ++	149 144 50.51
16	HUMPPIB MUSPPIA 21kDa (human), 19kDa (mouse) protein gene	1083 576 91.15	NA NA NA	558 558 92.83 97.31	NA NA NA	NA NA NA	558 558 92.83 97.31 ++																59 59 86.44 96.61 ++	88 90 51.69	70 70 85.71 94.29 ++	80 79 71.7
17	HUMKCHN MUSMK3A Voltage-gated potassium channel gene	2397 1994 83.25	NA NA NA	1572 1587 90.59 97.14	NA NA NA	NA NA NA	1572 1587 90.16 97.14 ++																				
18	HUMPCNA MMPCNAG Proliferating cell nuclear antigene (PCNA) gene	6340 4970 42.64	159 144 39.6	786 786 87.79 95.8	266 331 59.3	3713 2629 24.91	159 144 39.6	221 221 88.24 95.02 ++	708 815 39.4	98 98 84.69 97.96 ++	96 139 34.04	68 68 88.24 97.06 ++	938 252 9.412	195 195 88.72 96.92 ++	1885 1345 24.95	124 124 87.9 94.35 ++	86 78 63.41	80 80 87.5 93.75 ++	266 331 59.3								
19	HSU73304 MMU22948 CB1 cannabinoid receptor (CNR1) gene	5665 1654 84.46	63 62 76.8	1419 1422 90.42 97.25	NA 50 NA	>58 0 0	>58 - 0	63 62 76.8	1419 1422 90.32 97.25 ++	NA 50 NA																	
20	AF007876 MMATPB2 Na/K-ATPase beta 2 subunit gene	7894 7179 57.96	529 584 71.34	873 873 90.26 96.22	1854 NA NA	3790 3274 51.78	529 584 71.34	112 112 96.43 99.11 ++	2030 1737 47.46	129 129 90.7 95.35 ++	337 314 58.68	105 105 92.38 91.43 ++	106 101 62.8	206 206 87.86 96.12 ++	371 210 46.47	57 57 89.47 94.74 ++	841 800 60.09	99 99 85.86 93.94 ++	105 112 54.38	165 165 90.3 100 ++	1854 NA NA						

21	HUMNT3A	1029	NA	774	NA	NA	774																
	MMNT3	1284	NA	777	NA	NA	777																
	Neurotrophin-3 gene	90.28	NA	89.53	NA	NA	89.36																
				96.12			96.12																
							-+																
22	HSCKBG	4200	80	1146	297	1745	68	260	12	193	123	155	72	133	384	172	670	124	157	190	79	179	297
	MUSCRKNB	4521	92	1146	198	1503	77	256	15	193	186	155	77	133	169	172	585	124	147	190	83	179	198
	Creatine kinase B gene	60.38	58.14	89.62	64.65	33.90	55.17	31.01	74.07	87.05	44.66	88.39	61.74	91.73	27.85	88.37	32.99	93.55	40.13	92.63	25.93	87.15	64.65
				95.55						91.71		90.97		99.25		95.93		96.77		96.32		98.88	
										-+		++		++		++		++		++		++	
23	HUMACHRM4	2595	29	1437	NA	>771	>771	29	1437														
	MMM4ACHR	1707	NA	1440	NA	NA	NA	NA	1440														
	Muscarinic acetylcholine m4 receptor gene	79.96	NA	89.14	NA	NA	NA	NA	89.05														
				95.2					95.2														
									++														
24	HUMMHHSP2	2876	NA	1926	NA	NA	1926																
	MUSHSP7A2	3518	NA	1929	NA	NA	1929																
	MHC heat shock protein (HSP70-2) gene	80.88	NA	91.28	NA	NA	91.21																
				95.33			95.33																
							++																
25	HUMAPEXN	3730	332	957	258	1089	264	183	68	58	210	188	566	193	130	518	258						
	MUSAPEX	4042	129	954	165	962	72	183	57	55	217	188	460	193	102	518	165						
	APEX nuclease gene	49.71	32.54	86.1	36.41	36.87	20.83	38.8	64	70.8	44.5	88.83	30.6	86.53	49.14	86.87	36.41						
				94.04						62.07		89.36		97.93		97.88							
										++		++		++		++							
26	HUMGAD45A	5378	2327	498	547	1782	2327	44	486	102	223	238	1073	114	547								
	MUSGAD45	3100	NA	498	NA	1087	NA	44	89	102	94	238	904	114	NA								
	gadd45 gene	64.23	NA	89.76	NA	38.50	NA	95.45	17.04	87.25	27.13	87.82	50.58	93.86	NA								
				93.37				88.64		91.18		92.02		100									
								++		++		++		++									
27	HUMMHHSPHO	3330	NA	1926	NA	NA	1926																
	MUSHSC70T	2295	NA	1926	NA	NA	1926																
	Heat shock protein 70-HOM gene	81.39	NA	84.99	NA	NA	84.99																
				94.39			94.39																
							++																
28	HSHOX51	6305	1243	768	3126	541	1243	433	541	335	3126												
	MMU77364	7627	1140	753	572	558	1140	427	558	326	572												
	HOX 5.1 (human), HOXD4 (mouse) gene	46.77	69.9	88.15	23.9	72.79	69.9	87.91	72.79	90.47	23.9												
				92.19				90.76		94.03													
								-+		++													
29	HUMHISAC	1978	NA	660	NA	NA	660																
	MUSH1EH2B	3605	NA	660	NA	NA	660																
	Histone H1F4 (human), H1e (mouse) gene	75.38	NA	85.91	NA	NA	85.91																
				94.09			94.09																
							++																
30	HUMSPERSYN	7623	82	909	658	4169	82	167	431	121	332	93	2043	154	509	84	84	146	694	123	76	21	658
	MMSPERSYN	3915	75	909	277	1787	75	167	257	121	317	93	720	154	327		230		85	123	81	21	277
	Spermidine synthase gene	49.48	71.34	87.79	31.02	23.09	71.34	92.81	42.73	89.26	27.12	88.17	19.76	87.01	32.06	46.5	0	68.62	10.53	82.93	63.69	71.43	31.02
				93.73				97.01		99.17		100		93.51		92.86		90.41		85.37		85.71	
								++		++		++		++		++		-+		-+		++	

31	HSHIS10G	2530	NA	585	NA	NA	585																				
	MMU18295	2893	NA	585	NA	NA	585																				
	Histone H1(0) gene	66.84	NA	90.94	NA	NA	90.94																				
				94.87			94.87																				
							++																				
32	HSCFOS	3565	153	1143	786	1298	153	141	753	252	431	108	114	642	786												
	MMCFOS	3967	152	1143	846	1276	152	141	754	252	405	108	117	642	846												
	Cellular oncogene c-fos	81.68	74.75	87.93	86.27	67.11	74.75	89.36	73.39	83.73	56.94	90.74	64.07	88.79	86.27												
				93.7				97.87		86.9		100		94.39													
								++		++		++		+-													
33	HUMHGCR	2635	NA	1173	NA	NA	1173																				
	MUS5HT1B	2348	NA	1161	NA	NA	1161																				
	Serotonin 1B receptor gene	69.25	NA	88.41	NA	NA	88.86																				
				92.07			92.07																				
							++																				
34	HSODCG	9043	334	1386	342	5884	207	2856	110	105	17	102	283	174	206	173	258	135	130	82	176	84	1055	163	82	113	93
	MUSODCC	7100	425	1386	750	3948	196	1813	213	89	16	102	232	174	143	173	186	135	101	82	160	84	544	163	81	113	82
	Ornithine decarboxylase gene	51.94	60.54	85.71	51.8	41.19	70	51.55	42.72	51.5	60.61	85.29	44.66	86.21	40.11	89.6	29.73	86.67	42.42	84.15	50	86.9	28.27	86.5	39.26	83.19	36.57
				89.61								82.35		94.83		90.17		93.33		87.8		89.29		92.02		92.92	
												++		++		++		++		++		++		++		++	
35	HUMUDPCNA	4705	126	1338	1073	NA	126	1338	1073															215	640	145	342
	MUSGLCNACT	1894	NA	1344	NA	NA	NA	1344	NA															215	517	145	750
	Alpha-1, N-acetylglycosaminyltransferase	75.34	NA	84.6	NA	NA	NA	84.41	NA															87.91	16.25	77.93	51.8
				90.36				90.36																93.49		74.48	
								++																++		++	
36	HSU29185	35522	243	738	1606	12597	134	2622	99	9975	10	738	1606														
	MUSPRNPA	38418	155	765	1233	19923	47	2190	98	17733	10	765	1233														
	Prion protein Prp (human), Prnpa (mouse) gene	30.68	45.23	86.45	43.47	34.68	13.26	21.95	71.07	38.03	80	84.9	43.47														
				83.74								83.74															
												-+															
37	HUMGALTB	4286	27	1140	125	2620	27	82	308	170	232	76	95	49	125	130	153	57	162	123	305	133	103	84	327		
	MMU41282	4023	129	1083	97	1871	129	25	308	170	216	76	90	49	119	130	127	57	160	123	180	133	102	84	305		
	Galactose-1-phosphate uridyl transferase gene	60.08	16.67	81.67	54.05	46.24	16.67	35.51	51.3	88.24	53.12	82.89	63.78	75.51	69.67	94.62	61.43	92.98	67.7	80.49	33.4	90.23	63.41	92.86	53.48		
				85.79				25.61		97.06		86.84		73.47		96.92		100		85.37		90.23		96.43			
								-+		++		++		++		+-		X		X		X		-+			
38	HSU01212	3718	NA	492	NA	NA	492															155	810	81	125		
	MMU01213	3279	NA	492	NA	NA	492															155	264	81	97		
	Olfactory marker protein gene	49.19	NA	88.41	NA	NA	88.41															81.94	29.24	76.54	54.05		
				89.63			89.63															87.1		81.48			
							++															++		X			
39	HUMMIF	2167	97	348	116	284	97	108	189	173	95	67	116														
	MMU20156	920	92	348	120	344	92	108	201	173	143	67	120														
	Macrophage migration inhibitory factor	55.76	38.1	87.64	36.44	37.10	38.1	88.89	35.9	86.71	39.5	88.06	36.44														
				88.79				94.44		83.24		94.03															
								++		++		++															
40	AF049259	5698	47	1263	363	2938	47	495	1324	83	199	157	190	162	124	126	92	221	1009	19	363	-	-				
	MMU13921	4678	67	1314	249	2581	67	471	968	83	194	157	177	162	111	126	90	221	726	23	315	71	249				
	Keratin 13 gene	65.93	54.4	85.74	0	41.71	54.4	82.19	47.99	85.54	43.26	90.45	61.58	88.27	46.81	89.68	51.65	89.59	27.9	47.62	0	0	0				
				86.46				80.61		79.52		93.63		92.59		92.86		96.38		0							
								++		++		++		++		++		++		X							

41	HSH12	1391	NA	642	NA	NA	642															
	MUSHIS1A	1877	NA	639	NA	NA	639															
	Histone H1.2	60.03	NA	80.69	NA	NA	80.87															
	(human), H1.1			84.11			84.11															
	(mouse) gene						-+															
42	HSACTHR	1850	NA	894	NA	NA	894															
	MUSACTHR	1100	NA	891	NA	NA	891															
	Adrenocorticotrop	75.73	NA	83.89	NA	NA	84.03															
	ic hormone			88.26			88.26															
	receptor gene						+-															
43	AF027148	12825	172	963	622	763	172	630	489	79	274	254	622									
	MMMYOD1	2627	162	957	NA	767	162	627	439	79	328	251	NA									
	Myogenic	62.35	58.1	84.74	NA	48.06	58.1	89.1	47.63	79.75	48.84	76.44	NA									
	determining			87.85				94.76		68.35		76.77										
	factor 3 (MYOD1)							++		++		++										
44	HUMKER18	6520	47	1293	68	2376	47	417	741	83	337	157	572	165	85	126	262	224	379	121	68	
	MUSENDOBA	7879	58	1272	43	2412	58	396	792	83	274	157	639	165	111	126	275	224	321	121	43	
	Keratin 18	50.95	55.2	82.6	43.2	43.30	55.2	80.44	37.31	87.95	48.45	87.9	36.33	87.27	48.98	80.16	50.65	83.93	54.57	80.17	43.2	
	(human),			86.54				81.29		93.98		95.54		89.09		80.95		88.39		86.78		
	cytokeratin							++		++		+-		++		++		++		++		
45	HUMADRA	1521	NA	1353	NA	NA	1353															
	MUSALP2ADB	1454	NA	1353	NA	NA	1353															
	Alpha-2	85.21	NA	87.14	NA	NA	87.14															
	adrenergic			88.25			88.25															
	receptor gene						++															
46	HSMHCPU15	5833	46	660	18	3747	-	-	46	60	1853	68	1055	132	610	130	229	142		838	146	18
	MUSLMP2A	6101	356	642	297	3581	59	943	297		281		1310	132	213	130	382	99	44	733	253	
	Histocompatibility	41.66	16.6	62.12	5.079	15.8671	0	0	16.6	51.718	0	88.28	35.8	84.85	18.96	80	44.19	64.73	?	30	80.21	60.98
	comples			58.18						61.9		87.5		88.64		90		19.01			NA	
	(human),									X		++		++		++		++			++	
47	HSU72648	4850	NA	1386	NA	NA	1386															
	MUSADRA	2409	NA	1377	NA	NA	1377															
	Alpha-2 -C4-	78.83	NA	85.93	NA	NA	86.21															
	adrenergic			88.1			88.1															
	receptor gene						++															
48	HUMMK	4638	46	432	130	1295	45	296	1	76	160	168	122	162	717	26	130					
	MUSMKPG	2929	46	423	262	1417	45	481	1	76	117	159	116	162	703	26	262					
		54.39	73.88	86.34	44.4	23.12	73.3	18	100	82.89	40.43	86.24	34.45	90.12	19.44	88.46	44.4					
	Midkine gene			84.72						82.89		82.14		87.04		92.31						
										-+		++		++		++						
49	HSMYF4G	2804	NA	675	NA	258	NA	471	132	81	126	123	NA									
	MUSMYOGEN	4145	49	675	681	1041	49	471	514	82	527	122	681									
	Skeletal muscle-	78.42	NA	90.07	NA	26.49	NA	90.66	26.63	87.12	26.34	89.8	NA									
	specific,			94.67				96.82		85.19		92.68										
	myogenic gene							-+		++		++										
50	HUMHISAB	1314	NA	666	NA	NA	666															
	MMHISTH1	1943	NA	666	NA	NA	666															
	Histone H1 gene	78.61	NA	79.28	NA	NA	79.28															
	(H1F3, human)			87.84			87.84															
							++															

51	HUMOTNPI	1338	36	375	102	384	36	120	300	199	84	56	102														
	MUSOXYNEUI	2003	41	378	119	344	41	120	261	202	83	56	119														
	Prepro-oxytocin-	67.79	51.95	86.4	38.91	34.25	51.95	86.67	32.44	88.78	40.72	75	38.91														
	neurophysin I			87.2				90		91.96		64.29															
	gene							++		++		++															
52	HUMTKRA	13500	69	705	659	11534	69	66	110	32	1632	111	2374	94	6962	90	357	120	99	192	659						
	MUSTKM	2939	172	702	425	1476	172	66	81	32	492	111	225	94	222	90	151	120	305	189	425						
	Thymidine kinase	33.14	28.2	82.98	18.8	5.55	28.2	87.88	45.03	90.62	15.07	88.29	7.08	85.11	1.503	82.22	12.6	80.83	27.72	78.22	18.8						
	gene			86.38				90.91		93.75		91.89		89.36		86.67		92.5		75							
								++		++		++		++		++		++		++							
53	HUMADRBRA	3458	NA	1242	NA	NA	1242																				
	MMB2ARG	4928	NA	1257	NA	NA	1257																				
	Beta-2-adrenergic	72.79	NA	85.02	NA	NA	84.51																				
	receptor gene			87.2			87.2																				
							++																				
54	HUMMETIII	2167	NA	207	NA	1152	NA	31	248	66	904	110	NA														
	MUSMETIII	2649	56	207	138	1004	56	31	194	66	810	110	138														
	Metallothionein-III	50.99	NA	86.96	NA	41.26	NA	90.32	47.06	83.33	39.67	88.18	NA														
	gene			84.06				87.1		77.27		87.27															
								++		++		++															
55	HUMXRCC1G	37785	NA	1902	NA	30026	51	405	93	13889	111	6105	159	962	75	96	112	309	110	73	112	521	259	329	117	4453	94
	MUSXRCC1G	37349	NA	1890	NA	#REF!	51	431	93	11853	117	5681	159	524	75	95	112	357	110	77	100	169	259	237	117	2289	94
	XRCC1 DNA	26.55	NA	82.6	NA	#REF!	94.12	34.21	91.4	23.19	85.09	27.61	89.31	31.22	78.67	78.53	87.5	38.44	80	60	64.15	24.93	79.15	38.52	90.6	5.251	79.79
	repair gene			84.7			94.12		100		100		98.11		88		91.07		79.09		61.61		81.08		94.87		70.21
							-+		++		++		++		++		-+		++		++		++		++		++
56	HUMG0S24B	3889	58	981	706	810	58	24	810	957	706					156	133	488	55	94	140	1584	91	456	76	106	114
	MUSZPF36G	7493	59	960	770	677	59	24	677	936	770					139	130	293	55	88	137	1057	91	754	76	95	114
	GOS24 (human),	51.84	59.83	82.06	52.71	46.67	59.83	87.5	46.67	82.83	52.71					43.39	79.85	37.13	89.09	56.04	77.26	23.7	81.32	37.52	84.21	54.73	88.6
	Tristetraprolin zpf-			84.4				87.5		84.33							63.16		87.27		72.86		89.01		86.84		97.37
	36 (mouse) gene							++		++							++		++		++		++		++		++
57	HSAGL1	1138	37	429	109	257	37	95	117	205	140	129	109														
	MUSHBA	1441	32	429	102	256	32	95	122	205	134	129	102														
	Alpha-globin	59.23	27	79.72	51.2	22.96	27	72.63	25.1	79.02	21.17	86.05	51.2														
	germ line gene			85.31				72.63		86.34		93.02															
								++		++		++															
58	HSPGK2G	1911	NA	1254	NA	NA	1254																				
	MUSPGK2	2147	NA	1254	NA	NA	1254																				
	Testis-specific	68.71	NA	84.93	NA	NA	84.93																				
	PGK-2 gene			86.12			86.12																				
							++																				
59	HSU57623	9170	31	402	230	6712	31	73	3386	173	1889	102	1437	54	230												
	MMU02884	8765	41	402	238	6047	41	73	3441	173	1490	102	1116	54	238												
	Faty acid binding	43.83	44.4	82.34	62	39.71	44.4	89.04	38.85	84.97	39.83	76.47	41.6	75.93	62												
	protein (human),			85.07				90.41		84.97		82.35		83.33													
	MDGI (mouse)							++		++		++		++													
60	HSARYLA	3637	374	1524	99	1104	374	218	149	241	114	219	74	170	312	125	90	128	250	103	115	320	99				
	MMDNAASFA	4342	639	1521	NA	965	639	215	145	241	116	219	77	170	187	125	94	128	235	103	111	320	NA				
	Arylsulphatase A	62.41	45.21	82.15	NA	39.38	45.21	79.91	35.37	85.89	40	81.74	74.17	84.71	24.05	78.4	69.57	87.5	46.19	87.38	24.78	77.81	NA				
	gene			84.65				83.94		92.12		87.67		88.24		84		91.41		81.55		74.06					
								++		++		++		++		++		++		++		++					

61	S63168	1594	NA	810	NA	NA	810																				
	MUSCRP3A	1146	NA	804	NA	NA	804																				
	CCAAT/enhancer-	81.5	NA	85.06	NA	NA	85.38																				
	binding protein			85.19			85.19																				
	(human), C/EBP-						+-																				
62	HSHSP27	2496	41	600	139	841	41	364	723	64	118	172	139														
	MUSHSP25A	3058	59	630	128	723	59	375	596	66	127	189	128														
	HSP 27 (human),	50.44	11	83	42.7	31.71	11	82.81	27.29	87.69	58.78	74.79	42.7														
	HSP25 (mouse)			83.5				84.89		93.75		76.74															
	gene							++		++		++															
63	HUMROD1X	2841	15	1056	277	503	15	590	387	247	116	219	277														
	MUSROM1X	2787	15	1056	278	449	15	590	359	247	90	219	278														
	Rod outer	70	93.33	84.85	61.26	40.56	93.33	84.24	37	85.43	52.43	85.84	61.26														
	segment			84.38				83.39		88.66		82.19															
	membrane							--		++		++															
64	HUMSSTR3X	1413	NA	1257	NA	NA	1257																				
	MUSSSTR3A	1797	NA	1287	NA	NA	1287																				
	Somatostatin	81.03	NA	82.42	NA	NA	81.45																				
	receptor subtype			85.2			85.2																				
	3 gene						+-																				
66	HUMADAG	36741	95	1092	311	#REF!	95	33	15166	62	7052	123	2441	144	771	116	1237	128	1121	72	76	102	176	65	1438	130	616
	MMU73107	29807	92	1059	248	#REF!	92	33	11833	62	2689	123	2359	144	481	116	1024	128	552	72	81	102	221	65	1764	130	471
	Adenosine	34.05	48.1	78.39	35.7	#REF!	48.1	84.85	19.36	83.87	1.643	81.3	33.83	86.11	34.82	82.76	36.62	78.91	27.5	70.83	48.41	78.43	46.85	78.46	38.41	83.08	24.1
	deaminase (ADA)			80.22				90.91		87.1		75.61		91.67		77.59		75		83.33		85.29		73.85		92.31	
	gene							++		++		++		++		++		++		++		++		++		++	
65	HSPNMTB	3799	23	849	83	1065	23	202	951	208	114	439	83										103		448	14	311
	MUSPNMT	3144	NA	888	NA	578	NA	235	470	208	108	445	NA										84	16	495	232	
	Phenylethanolam	55.12	NA	82.1	NA	23.44	NA	70.02	21.82	83.17	36.94	83.94	NA										60.96	0	32	71	35.7
	ine N-methylase			84.45				75.74		86.54		87.47											58.25			NA	
	gene							-+		++		++											X			X	
67	HUMSMPD1G	5588	86	1890	369	2113	86	312	465	773	1061	172	229	77	202	146	156	410	369								
	MMASM1G	4775	NA	1884	NA	1617	NA	306	383	773	512	172	240	77	354	146	128	410	NA								
	Acid	61.09	NA	81.27	NA	41.98	NA	65.05	49.29	83.31	36.24	90.7	44.78	87.01	36.69	85.62	61.97	83.66	NA								
	sphingomyelinas			82.06				54.81		87.32		92.44		89.61		90.41		84.15									
	e gene (SMPD1,							-+		++		++		++		++		++									
68	HSIGF2G	8837	NA	394	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	2661	6	157		1701		237	NA				
	MMU71085	27874	2605	543	3044	12842	223	1855	234	5657	929	1430	1119	882	94	1566	6	157	1205	149	247	237	3044				
	Insulin-like	54.65	NA	83.76	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	27.11	66.67	86.62	NA	NA	NA	81.86	NA				
	growth factor II			45.69														85.99		NA		18.99					
	gene																	-+		NA		++					
69	HUMNUCLEO	10942	111	2124	283	6912	111	18	960	117	1163	478	670	198	104	87	184	142	1005	125	605	124	459	158	514	124	236
	MMNUCLEO	11478	148	2124	NA	6480	148	18	982	117	656	496	215	186	98	87	177	142	907	125	1163	124	450	155	492	124	190
		43.68	55.6	84.42	NA	28.79	55.6	88.89	32.34	82.05	18.8	80.7	18.31	80.73	54.46	87.36	50.42	80.28	17.05	89.6	32.58	83.06	24.64	84.98	25.84	91.13	41.31
	Nocleonin gene			82.34				100		87.18		75.31		66.67		86.21		69.72		96		77.42		79.75		84.68	
								++		++		++		++		++		++		++		++		++		++	
70	HUMCOX5B	2593	NA	390	83	1628	-	103	876	74	200	100	552	113	83					134	493	127	386	215	133	77	283
	MUSCYTCOVB	2540	49	387	76	1336	49	100	521	74	166	100	649	113	76					122	704	127	331	224	115	77	NA
	Cytochrome c	47.64	0	84.36	69.18	40.04	0	82.76	29.63	87.84	48.63	87	53.46	82.3	69.18					85.94	45.45	88.98	52.16	86.56	38.71	89.61	NA
	oxidase Vb			80.77				75.73		89.19		87		74.34						82.84		94.49		99.07		93.51	
	subunit gene							++		++		++		++						++		++		++		++	

71	S45332	8647	NA	1527	NA	4686	NA	115	856	136	987	176	80	158	483	154	84	88	2101	88	95	612	NA				
	MMERYPR	6561	45	1524	163	2852	45	115	977	133	333	176	85	158	606	154	78	88	679	88	94	612	163				
	Erythroproietin	45.21	NA	81.6	NA	26.98	NA	81.74	36.55	84.76	15.61	77.27	44.85	77.22	28.65	86.36	66.67	79.55	24.17	93.18	62.43	80.88	NA				
	receptor gene			80.75				78.26		79.41		80.11		75.95		89.61		85.23		92.05		78.43					
								++		++		++		++		++		++		++		++					
72	HUMINSPR	3943	59	333	NA	966	42	179	17	187	787	146	73														
	MMINSIIG	2408	63	333	>39	606	46	118	17	186	488	147	>39														
	Alpha-type insulin	29.9	33.1	81.68	NA	20.95	22.7	29.6	58.8	79.36	18.98	84.64	NA														
	(human),			81.08						80.21		82.19															
	preproinsulin									++		++															
73	HUMHST	6616	238	621	1855	1155	238	340	617	104	538	177	1855														
	MMKFGF	3989	117	609	2301	828	117	328	399	104	429	177	2301														
	transforming	57.28	26.48	80.35	46.78	34.06	26.48	78.14	33.46	85.58	34.75	84.18	46.78														
	protein (hst,			79.23				70.59		86.54		91.53															
	human),							++		++		+-															
74	HUMPLPSPC	3409	25	594	428	>1600	25	42	698	159	345	123	223	111	334	159	428	NA	NA								
	MUSPSPC	3633	27	582	185	2329	27	42	1229	159	334	123	237	111	328	147	9	201	176								
	Pulmonary	55.12	26.92	78.28	0.326	NA	26.92	83.33	40.69	83.65	53.31	79.67	46.52	84.68	48.04	68.63	NA	NA	NA								
	surfactant protein			78.79				85.71		90.57		78.05		78.38		66.04											
	(SP-C) gene							++		++		++		+-		++											
75	HSCOSEG	3521	NA	495	60	1550	NA	120	1376	202	174	173	60														
	MUSVASNEU	3494	32	507	52	1349	32	132	1145	202	204	173	52														
	Vasopressin-	42.93	NA	84.44	30.36	36.40	NA	80.16	34.91	91.58	48.15	76.3	30.36														
	neurophysin			81.21				82.5		93.56		65.9															
	precursor gene							-+		++		++															
76	HSB3A	3683	197	1227	636	>1025	197	1205	1025	22	636	NA	NA														
	MMB3A	3438	182	1203	708	809	182	1163	463	40	28	346	680														
		62.71	54.88	78	NA	NA	54.88	80.41	22.72	16.13	NA	NA	NA														
	Beta-3-adrenergic			80.2				81.41		13.64																	
	receptor gene							+-		X																	
77	HUMLORI	3321	62	951	236	1188	39	1188	23	951	236																
	MMU09189	6530	70	1446	235	1091	47	1091	23	1446	235																
	Loricrin gene	63.26	63.64	76.13	68.37	53.18	65.12	53.18	60.87	60.41	68.37																
				79.18						79.18																	
78	HSU37055	9980	76	2136	80	2474	76	52	697	148	80	113	77	115	77	137	79	121	144	119	202	169	120	131	97	103	127
	MUSHEPGFA	6751	94	2151	65	2303	94	52	613	148	92	113	84	115	84	137	81	121	81	119	143	196	159	131	78	103	122
	Hepatocyte	65.87	75.29	79.78	46.9	49.80	75.29	76.92	47.48	83.78	60.47	76.11	65.84	80.87	59.63	83.21	52.5	76.03	35.56	79.83	42.9	71.23	52.33	75.57	38.86	83.5	32.13
	growth factor-like			78.51				69.23		81.08		76.99		78.26		83.21		76.86		83.19		71.01		75.57		84.47	
	protein gene							++		++		++		++		++		++		++		++		++		++	
79	HSH11	1034	NA	648	NA	NA	648									137	89	36	88	121	175	78	127	147	81	107	95
	MUSH1X	1781	NA	642	NA	NA	642									137	88	36	79	109	161	78	119	147	80	107	98
		72.34	NA	71.76	NA	NA	72.09									75.91	66.67	86.11	50.3	71.3	48.81	85.9	43.9	80.95	68.32	85.05	67.36
	H1.1 (human), H1			79.17			79.17									76.64		91.67		76.86		84.62		81.63		78.5	
	(mouse) gene						++									++		++		++		++		++		++	
80	HSU66875	1569	75	294	28	269	75	73	101	137	168	84	28											140	119	162	80
	MMU63716	2324	NA	294	NA	263	NA	73	186	137	77	84	NA											140	141	162	65
	COX6AH	55.77	NA	79.25	NA	30.96	NA	82.19	39.02	75.91	26.12	82.14	NA											82.14	60	83.95	46.9
	(human),			77.55				73.97		72.26		89.29												68.57		83.33	
	coxVIaH (mouse)							++		++		++												++		++	

81	HSINT2	11608	NA	720	337	7937	NA	220	2289	104	5648	396	337																		
	MMINT2	8283	864	738	1454	3543	864	220	1743	104	1800	414	1454																		
	Int-2 proto-oncogene	32.21	NA	79.86	22	8.29	NA	87.27	27.43	82.69	0.537	73.33	22																		
				82.5				94.09		92.31		73.48																			
								++		++		+-																			
82	HSBGL3	2052	50	444	132	980	50	92	130	223	850	129	132																		
	MUSHBBMAJ	6532	52	444	2135	769	52	92	116	223	653	129	2135																		
	Beta-globin major gene	49.9	78.4	81.76	8	36.18	78.4	83.7	56.1	82.51	33.13	79.07	8																		
				79.73				78.26		82.06		76.74																			
								++		++		++																			
83	HUMALIFA	7614	64	609	3073	2424	64	19	1731	179	693	411	3073																		
	MUSALIFA	8757	171	612	>5086	2117	171	19	1560	182	557	411	>5067																		
	Leukemia inhibitory factor (LIF) gene	55.79	47.66	80.79	NA	NA	47.66	89.47	NA	86.98	42.88	77.37	NA																		
				79.8				94.74		90.5		74.45																			
								++		++		++																			
84	HUMFABP	5204	NA	399	NA	2661	67	1194	173	1023	108	444	51																		
	MUSFABPI	5039	NA	606	NA	3467	103	1256	173	1792	108	419	222																		
	Intestinal fatty acid binding protein gene	43.86	NA	80.2	NA	34.10	64.71	34.53	81.5	35.31	81.48	30.13	26.37																		
				60.15			13.43		79.77		66.67		41.18																		
							-+		++		++		++																		
85	HUMLYTOXBB	6305	8	735	151	974	8	162	396	46	183	72	395	455	151																
	MMU16984	6914	10	921	155	704	10	162	366		316		338	443	155																
	Lymphotoxin-beta gene	39.52	66.67	78.21	5.9	#######	66.67	85.8	50.13	54.3	?	76.4	31.4	80.4	5.9																
				82.43				85.19		76.5		80		74.5																	
								-+		X		-+		++																	
86	HUMLYL1B	4569	403	804	291	2197	345	1307	58	296	87	92	803	416	291																
	MMLYL1	3678	372	837	-	1458	348	652	24	333	91	90	715	414	-																
	LYL-1 protein gene	51.52	53.16	78.23	0	29.72	58.87	25.42	4.878	70.27	73.03	87.91	32.02	79.04	0																
				76.87						73.99		88.04		76.44																	
										-+		++		++																	
87	HUMANFA	2710	NA	456	NA	1215	123	122	327	1093	6																				
	MUSANF	1983	NA	459	NA	490	120	103	327	387	12																				
	Atrial natriuretic factor gene	63.94	NA	79.82	NA	24.41	74.9	54.22	82.57	21.08	50																				
				78.95			63.41		84.4		50																				
							++		++		X																				
88	HUMG0S19A	4102	83	279	413	1108	83	73	688	115	420	91	413																		
	MMSCIMIP	1988	80	279	400	719	80	76	490	112	229	91	400																		
	MIP1/SCI murine gene and human homolog	55.84	69.9	77.42	56.1	33.27	69.9	79.19	38.71	77.53	24.35	75.82	56.1																		
				75.27				82.19		75.65		69.23																			
								++		++		++																			
89	HUMALPI	5291	50	1587	886	1360	50	67	82	117	114	116	203	175	76	173	242	135	95	73	130	135	84	192	223						
	MUSIAP	5293	NA	1680	NA	1225	NA	67	84	117	116	116	117	175	78	173	261	135	81	73	129	135	73	192	200						
	Intestinal alkaline phosphatase (ALP1, human),	50.95	NA	78.45	NA	38.55	NA	88.06	50.6	84.62	50.43	82.76	34.38	85.14	55.84	80.92	34.99	79.26	30.68	67.12	33.2	77.78	30.57	80.73	46.34						
				78.07				94.03		84.62		85.34		82.29		86.71		82.22		57.53		68.89		84.38							
								-+		++		++		++		++		++		++		++		++							
90	HUMFPR1A	6931	136	1053	166	4766	74	1559	51	3207	11	1053	166									117	111	287	886						
	MUSNFORREC	1524	NA	1095	NA	NA	NA	NA	NA	NA	NA	1095	NA									114	86	383	NA						
	N-formyl peptide receptor gene	56.36	NA	77.78	NA	NA	NA	NA	NA	NA	NA	76.26	NA									67.53	24.37	62.09	NA						
				77.49								77.49										64.1		70.03							
												++										++		+-							

91	HSSPRO	5296	NA	1437	NA	1390	64	92	120	75	345	78	140	256	157	197	153	459	345	233	113											
	MMVITRO	5004	NA	1437	NA	1460	64	89	120	78	342	79	140	167	157	109	153	725	351	213	110											
	S-protein (human), vitronectin	56.83	NA	76.83	NA	45.94	89.06	53.04	82.5	71.9	61.72	58.6	90	31.68	85.99	33.33	86.27	44.09	73.28	60.54	78.92											
				73.9			79.69		75		54.78		94.29		87.9		86.27		69.57		79.65											
							++		++		++		++		++		++		++		++											
92	HSGCSFG	2960	35	624	853	862	35	40	176	164	378	108	144	147	164	165	853															
	MMGCSFG	3054	36	627	NA	953	36	40	84	173	348	108	246	147	275	159	NA															
	Granulocyte colony-stimulating factor	53.89	53.5	75.64	NA	32.93	53.5	75	24.62	70.03	32.78	86.11	38.97	76.19	36.9	73.46	NA															
				72.12				52.5		64.02		86.11		83.67		65.45																
								++		+-		++		++		X																
93	HUMTNFBA	2140	170	618	NA	621	162	288	8	99	86	106	247	413	NA																	
	MMTNFBG	3219	116	609	574	644	107	337	9	96	83	100	224	413	574																	
	Tumor necrosis factor-beta gene	54.02	48.25	77.99	NA	55.00	46.84	54.08	70.59	69.74	74.56	69.9	49.26	82.81	NA																	
				71.84						57.58		56.6		79.18																		
										-+		++		++																		
94	HSU16720	8868	NA	537	1034	3241	NA	165	866	60	290	153	1010	66	1075	93	1034															
	MUSIL10Z	7207	65	537	702	3820	65	165	887	60	305	153	1007	66	1621	93	702															
	Interleukin 10 (IL10) gene	48.37	NA	80.07	58.1	42.31	NA	73.33	38.22	76.67	45.04	84.97	48.69	89.39	38.87	79.57	58.1															
				73.18				61.82		70		82.35		90.91		67.74																
								++		++		++		++		++																
95	HUMCP21OH	4042	NA	1488	NA	1224	202	97	90	282	155	107	102	88	102	101	87	169	201	200	179	83	104	97	266							
	MUS21OHA1	3307	NA	1464	NA	1159	202	68	90	352	143	105	96	86	102	88	87	110	201	198	170	68	104	84	269							
	21-hydroxylase B (human), 21-hydroxylase A	59.69	NA	73.92	NA	33.48	78.71	27.88	73.33	31.86	75.17	21.7	74.75	65.52	68.63	62.43	75.86	29.39	73.63	12.06	73.93	59.6	78.85	26.52	72.52							
				70.56			78.71		70		73.55		76.47		47.06		68.97		71.64		70.39		72.12		68.8						
							++		++		++		X		++		++		++		++		++		++						
96	HUMMIS	3100	10	1683	109	942	10	412	592	143	172	109	88	160	90	859	109															
	MMAMH	2870	8	1668	NA	735	8	403	372	143	177	109	87	160	99	853	NA															
	Anti-Mullerian hormone gene	56.52	50	72.96	NA	25.14	50	65.52	15.56	72.03	36.68	76.15	54.86	70.62	37.04	77.34	NA															
				71.48				56.8		67.13		74.31		65.62		79.98																
								++		++		++		++		++																
97	HUMAPOE4	5515	67	954	142	2431	44	757	23	43	1093	193	581	718	142																	
	MUSAPE	4856	65	936	109	1675	42	759	23	43	539	169	377	724	109																	
	Apolipoprotein gene	37.75	68.18	76.62	33.47	27.32	65.12	27.97	73.91	81.4	23.53	74.03	33.61	77.95	33.47																	
				71.07						69.77		65.28		72.7																		
										-+		++		++																		
98	HUMREGB	4251	74	501	193	2153	28	301	46	64	635	119	307	138	715	112	195	68	193													
	MUSREGI	3756	71	498	200	1915	25	277	46	61	521	119	293	138	651	112	173	68	200													
	Regenerating protein gene	38.79	71.59	76.85	37	36.36	64.2	43.25	76.09	73.6	37.89	74.79	51	82.61	22.55	75.89	48.37	75	37													
				71.86						56.25		75.63		80.43		75		57.35														
										++		++		++		++		++														
99	HUMPROLA	1404	NA	1002	NA	NA	1002																									
	MUSPROL	1413	NA	1005	NA	NA	1005																									
	Cathepsin L gene	67.88	NA	76.45	NA	NA	76.33																									
				71.56			71.56																									
							++																									
100	HSU29874	6155	>80	708	21	>5347	>80	33	1018	111	344	54	224	144	2297	139	1249	179	75	48	21	>140										
	MMU44024	4799	63	699	NA	3181	63	33	240	122	432	46	605	144	344	144	1470	189	90	21	NA	NA										
	Flt3 ligand gene	30.4	NA	68.64	NA	NA	NA	93.94	19.71	64.38	37.89	78	36.43	75.69	7.8	77.03	29.86	61.41	46.06	28.99	NA	NA										
				67.37				90.91		70.27		72.22		75		73.38		62.01		18.75												
								++		X		++		++		++		++		X												

101	HSA6693	3448	NA	510	NA	NA	510																				
	MUSSER1	3366	NA	693	NA	NA	693																				
	Ultra high sulfur protein gene	28.55	NA	79.02	NA	NA	67																				
				59.41			59.41																				
							--																				
102	HUMIL2RGA	4038	14	1110	145	2696	14	115	378	154	208	185	208	140	763	163	532	97	252	70	355	186	145				
	MMU21795	5267	24	1110	729	2210	24	115	190	154	216	185	190	143	636	163	454	97	187	67	337	186	729				
	IL2RG (human), common cytokine receptor gamma	57.36	63.2	79.73	21.3	51.03	63.2	70.43	40.14	81.82	60.38	78.92	72.86	74.91	36.45	80.37	57.4	81.44	46.47	83.21	69.36	85.48	21.3				
				69.73				44.35		74.03		71.35		64.29		64.42		71.13		77.14		85.48					
								-+		++		++		++		++		++		++		++					
103	HUMCRPGA	2480	NA	675	NA	278	NA	61	278	614	NA																
	MMCRPG	2140	84	678	921	213	84	64	213	614	921																
	CRP gene for C-reactive protein	45.75	NA	75.7	NA	41.55	NA	67.2	41.55	76.38	NA																
				69.78				54.1		71.34																	
								++		++																	
104	HSBCDIFFI	3230	NA	405	NA	1259	NA	144	208	33	945	129	106	99	NA												
	MMIL5G	6727	43	402	1089	2783	43	141	829	33	1875	129	79	99	1089												
	B cell diff. Factor I (human), eosinophil diff.	53.13	NA	75.31	NA	34.01	NA	68.07	23.53	75.76	36.74	84.5	30.27	74.75	NA												
				70.37				60.42		72.73		74.42		78.79													
								-+		++		++		++													
105	HUMTHY1A	2806	NA	486	NA	1010	37	483	336	527	113																
	MUSTHY1GC	3257	NA	489	NA	976	37	590	339	386	113																
	Thy-1 (human), Thy-1.2 (mouse) gene	55.49	NA	75.51	NA	36.92	83.78	38.96	71.11	35.05	84.96																
				69.14			72.97		63.39		84.96																
							++		++		++																
106	HSUPA	7258	119	1296	929	4043	88	306	31	57	417	28	146	108	603	175	193	92	157	220	221	149	665	141	346		
	MUSUPAA	9950	101	1302	909	4382	71	318	30	57	484	31	137	108	626	175	396	92	143	223	220	149	574	141	306		
	Urokinase-type plasminogen activator protein	56.03	47.03	76.39	59.6	43.65	42.8	44.6	59.02	71.93	43.51	47.46	58.66	78.7	25.71	81.71	41.77	84.78	59.33	74.04	58.5	71.14	55.37	76.6	47.24		
				68.06						52.63		21.43		66.67		70.29		71.74		69.55		62.42		65.96			
										++		++		++		++		++		++		++		++			
107	HUMSAP01	1394	96	672	156	115	96	64	115	608	156											149	989	177	929		
	MUSSAPRB	1350	152	675	141	110	152	67	110	608	141											149	1178	177	909		
	Serum anyloid P component gene	61.48	31.5	74.11	39.1	66.67	31.5	79.39	66.67	73.36	39.1											79.87	37.56	74.58	59.6		
				68.75				65.62		69.08												74.5		76.27			
								++		++												++		-+			
108	HUMPAP	4497	60	528	219	1939	26	288	34	76	560	119	187	138	626	127	278	68	219								
	D63360	4292	31	528	200	1823	26	289	5	76	518	119	170	138	586	127	260	68	200								
	PAP (human), reglilbeta/PAP (mouse) gene	43.08	19.78	73.67	43.44	38.19	30.77	33.97	5.128	73.68	37.29	75.63	34.73	78.99	36.63	66.93	50.19	72.06	43.44								
				68.75						78.95		60.5		73.91		66.14		66.18									
										-+		++		++		++		++									
109	HUMIL1B	7824	87	810	598	5512	72	463	15	47	565	52	1988	202	545	165	1235	131	716	213	598						
	MMIL1BG	7100	87	810	431	5211	72	721	15	47	533	49	1540	202	547	171	1150	131	720	210	431						
	Interleukin 1-beta gene	53.52	72.45	77.04	46.8	43.1002	68.1	45.61	93.33	68.09	57.01	77.23	41.95	73.27	51.47	69.64	39.16	83.21	34.12	84.63	46.8						
				67.78						51.06		63.46		56.44		60		84.73		78.87							
										-+		++		++		++		++		++							
110	HUMCAPG	3734	28	768	84	1819	28	55	756	148	454	136	175	255	434	174	84										
	MUSCATHG	3438	28	786	57	1688	28	55	633	148	487	136	176	255	392	192	57										
	Cathepsin G gene	51.92	NA	72.27	12.8	41.64	75	76.36	42.19	83.11	38.47	77.94	31.91	66.67	47.94	62.3	12.8										
				66.41				70.91		79.05		75		61.18		55.17											
								++		++		++		++		X											

111	HUMCTLA1	4505	33	744	113	2346	33	55	1043	148	455	136	205	261	643	144	113				
	MUSSPCTLS	4348	NA	744	NA	1972	NA	55	854	148	370	136	134	261	614	144	NA				
	CTLA1 (human), CCPI (mouse) gene	54.67	NA	75 66.94	NA	48.10	NA	63.64 43.64	49.45	79.05 72.97	41.7	80.15 77.21	40.12	71.26 57.47	52.98	77.08 77.08	NA				
								++		++		++		++		++					
112	HSLACTG	3310	26	429	272	1635	26	133	647	159	489	76	499	61	272						
	MUSALCALB	3045	35	432	289	1549	35	136	301	159	468	76	780	61	289						
	Alpha-lactalbumin gene	54.25	52.46	74.13 66.43	61.32	39.22	52.46	70.63 60.9	32.7	72.96 69.81	53.08	81.58 71.05	34.09	73.77 63.93	61.32						
								++		++		++		++							
113	HUMOSTP	10881	104	945	424	6129	90	1083	14	54	109	39	2941	81	265	42	1039	324	692	405	424
	MMOESTEOP	5782	196	885	431	3354	NA	NA	196	54	98	36	1075	81	496	42	998	282	687	390	431
	Osteopontin gene	54.77	7.333	66.98 64.44	66.43	28.29	NA	NA	10.48	75.93 66.67	54.11	69.33 76.92	24.45	81.48 77.78	44.15	88.1 85.71	52.53	60.73 60.19	42.35	70.19 61.48	66.43
										++		++		++		++		-+		++	
114	HSCD14G	1570	105	1128	123	88	105	3	88	1125	123										
	MMCD14	2404	113	1101	NA	95	113	3	95	1098	NA										
	DCD14 differentiation antigen gene	68.34	55.96	70.21 65.16	NA	65.57	55.96	66.67 100	65.57	71.08 65.07	NA										
								X		++											
115	HSGAPIGNA	2609	6	348	215	1459	6	75	1027	208	432	65	215								
	MMU60528	5416	36	351	200	1172	36	75	774	211	398	65	200								
	gap-I (human), guanylin precursor	47.6	19.05	74.14 65.52	29.88	40.34	19.05	77.33 68	39.87	69.21 59.13	41.45	84.62 83.08	29.88								
								++		++		++									
116	HSBGPG	1675	NA	297	NA	634	NA	64	257	33	176	70	201	130	NA						
	MUSOGC	949	48	288	122	491	48	64	144	33	141	58	206	133	122						
	Bone gla protein (human), osteocalcin	41.41	NA	68.69 61.62	NA	23.65	NA	68.75 60.94	22.44	72.73 72.73	42.9	68.75 55.71	8.354	69.96 62.31	NA						
								++		++		++		++							
117	HSAPOAIA		39	804	56	972.0	18	197	21	43	187	157	588	604	56						
	MUSAICIIIA		NA	798	NA	NA	NA	NA	NA	43	116	157	545	598	NA						
	Apolipoprotein gene		NA	74.7 63.39	NA	NA	NA	NA	NA	81.4 56.3	46.2	81.7 72.2	24.7	72.4 61.6	NA						
										++		++		++							

Appendix D

Whole Genome Assembly Using Combined Shotgun and Clone-by-Clone Sequencing (WGSCC Sequencing)

Below we describe a procedure by which a large genome could be sequenced and assembled using a specific combination of whole genome shotgun and clone-by-clone data (WGSCC data).

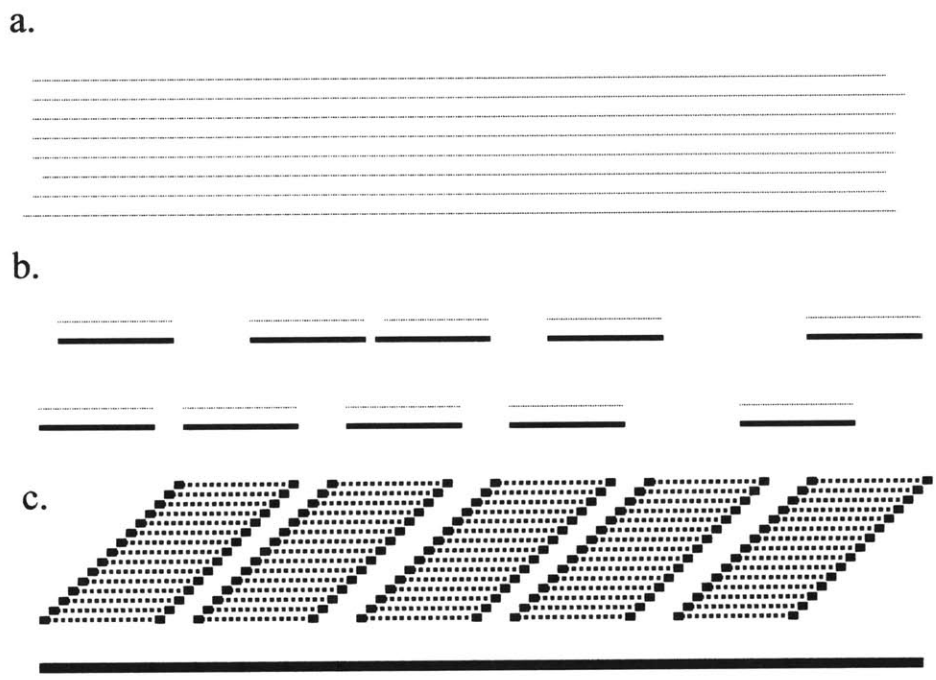

Figure D.1. a. Whole genome shotgun reads covering the genome; **b.** Clone-by-clone light shotgun; **c.** deep library of end-sequenced BACs.

The procedure assumes the existence of the following data:

1. Plasmid whole genome shotgun paired reads (WGS reads) covering the genome at depth Kx where K is a constant < 10 (Figure D.1a).
2. A random collection of BAC clones covering the genome to Lx, each clone being sequenced to a small Mx coverage (Figure D.1b).

3. A deep library of end-sequenced BAC clones (of depth at least 15, Figure D.1c).

Typical values for K, L, and M would be 7, 2, and 0.5, respectively. The depth of the BAC library could be around 20 deep. For these values, the total cost measured in equivalent whole genome shotgun coverage depth, would be:

1. 7x, because we have a depth K = 7 of shotgun reads.
2. LMx = 1x.
3. Assuming that end-sequencing a BAC is equally expensive as twice the end-sequencing of a plasmid insert, we get 2 times 20 (depth) times 4 reads / 150kb. Letting each read be roughly 500 long, this gives a total of 0.25x cost.

Therefore the total cost of obtaining this initial data is 8.25x.

Then, a majority of the genome would be covered with BAC clones that are obtained in step 2 above, and are each sequenced to 0.5x. This would correspond to having roughly 150 reads per clone, or one read per 1,000 bases. This data can be used to assemble each of the light-sequenced BAC clones. Below we describe the algorithm:

Algorithm: assembling a BAC clone using a set of reads $r_1, \ldots, r_n$ from the clone and a Kx (K roughly = 7) coverage of the genome with random shotgun.

1. Form adjacency graph of reads for the above reads, plus the WGS reads (refer to the first steps of the *ARACHNE* algorithm, Chapter 3).
2. Find in this graph the pairwise shortest paths for reads $r_1, \ldots, r_n$ where distance is measured by the overlap shifts (refer to Chapter 3).
3. Order and orient $r_1, \ldots, r_n$ using these shortest paths. Report the existense of ambiguities, if any.
4. The above steps result in one or more contigs covering the entire clone. If any ambiguities exist in step 3, there is possibly more than one contig. Order and orient these contigs using the plasmid paired reads of WGS.
5. If more than one contig, fill in the regions between contigs using shortest paths in the overlap graph.
6. For the resulting assembly of the clone, determine if any regions exist that are the result of overcollapsing WGS reads from two or more copies of a repeat. Do that heuristically by

measuring the density of reads covering specific subregions (high density means higher probability of overcollapsing (Myers et al. 2000)).

We conjecture that the full sequence of most of these clones can be reliably obtained using the above procedure. Moreover, whenever there is ambiguity, or high probability of obtaining lower quality consensus sequence due to the presence of repeats, the algorithm reports this in steps 3 and 6. We conjecture that the majority of the clones will be assembled with virtually no ambiguities, and with reliable resulting consensus sequence. Some clones will be assembled with lower accuracy guarantees, reported in step 6. And few clones will not be assembled successfully.

The assembled clones can form islands of fully sequenced genome, separated with *oceans* of unassembled genome. The library of end-sequenced BACs could then be employed to walk from these islands similarly to the way described in chapter 2. In this case though, it would no longer be as important to obtain a minimal overlapping clone from the library. The library clones selected for walking would only be sequenced to 0.5x depth resulting in negligible wastage due to overlaps with already assembled islands.

The whole scheme as described above involves performing a total sequencing of around 8.5x. Therefore there is considerable slack up to the total maximum of 10x, to devote additional sequencing of specific clones in "difficult" regions of DNA. For instance, the hardest 5% of the genome consisting of repeats that are hard to resolve and assemble with 99.99% accuracy with whole genome shotgun could be sequenced clone-by-clone (after failure to assemble) to depth 5x. The total additional sequencing this step would involve is only 0.25x, well within the allowable slack. Then, the 5x sequencing of these clones together with the 7x whole genome shotgun should enable to yield the best possible quality of assembly in these regions, similar to 10x clone-by-clone sequencing.

To summarize, the procedure runs as follows. (1) Obtain WGS reads, random BAC clone-by-clone reads, and deep BAC library. (2) Assemble the lightly sequenced clones as described above, resulting in islands of sequenced genome. After this step most of the islands should be assembled with high-fidelity sequence, with some marked regions of lower fidelity. (3) Walk using the end-sequenced clone library, covering the entire genome and potentially increasing the fidelity of some regions. After this step, roughly 8.5x sequencing has been used. The result is a full genome sequence, with some parts marked as lower fidelity. The remaining 1.5x sequencing that we are allowed to perform can be used to "finish" these parts. This should be easy given that virtually all parts of poor quality should be "captured" by lightly sequenced clones, and by library clones.

The above procedure can be varied considerably. For instance, it is not clear which is the optimal combination of WGS data (unpaired reads, plasmid paired reads, cosmid paired reads). A

variation of the above method could use a higher depth of coverage of the genome with lightly shotgun sequenced BAC clones, with each clone sequenced to lower depth. For instance at an extreme a 15x coverage of the genome could be obtained with BACs, each being sequenced to depth 0.1x (around one read per 5kb, or ~30 reads total) for a total sequencing of 1.5x. The same clones could also be end-sequenced. That would result in minimizing the need for walking in order to close gaps of unassembled genome, and thus may be done more in parallel. Or, a library of shorter BACs and/or cosmids could be also employed, in order to do more targeted finishing by going into very deep sequencing in small highly problematic regions.

BIBLIOGRAPHY

Adams, M.D., S.E. Celniker, R.A. Holt, C.A. Evans, J.D. Gocayne, P.G. Amanatides, S.E. Scherer, P.W. Li, R.A. Hoskins, R.F. Galle, R.A. George, S.E. Lewis, S. Richards, M. Ashburner, S.N. Henderson, G.G. Sutton, J.R. Wortman, M.D. Yandell, Q. Zhang, L.X. Chen, R.C. Brandon, Y.-H.C. Rogers, R.G. Blazej, M. Champe, B.D. Pfeiffer, K.H. Wan, C. Doyle, E.G. Baxter, G. Helt, C.R. Nelson, G.L. Gabor Miklos, J.F. Abril, A. Agbayani, H.-J. An, C. Andrews-Pfannkoch, D. Baldwin, R.M. Ballew, A. Basu, J. Baxendale, L. Bayraktaroglu, E.M. Beasley, K.Y. Beeson, P.V. Benos, B.P. Berman, D. Bhandari, S. Bolshakov, D. Borkova, M.R. Botchan, J. Bouck, P. Brokstein, P. Brottier, K.C. Burtis, D.A. Busam, H. Butler, E. Cadieu, A. Center, I. Chandra, J.M. Cherry, S. Cawley, C. Dahlke, L.B. Davenport, P. Davies, B. de Pablos, A. Delcher, Z. Deng, A. Deslattes Mays, I. Dew, S.M. Dietz, K. Dodson, L.E. Doup, M. Downes, S. Dugan-Rocha, B.C. Dunkov, P. Dunn, K.J. Durbin, C.C. Evangelista, C. Ferraz, S. Ferriera, W. Fleischmann, C. Fosler, A.E. Gabrielian, N.S. Garg, W.M. Gelbart, K. Glasser, A. Glodek, F. Gong, J.H. Gorrell, Z. Gu, P. Guan, M. Harris, N.L. Harris, D. Harvey, T.J. Heiman, J.R. Hernandez, J. Houck, D. Hostin, K.A. Houston, T.J. Howland, M.-H. Wei, C. Ibegwam, M. Jalali, F. Kalush, G.H. Karpen, Z. Ke, J.A. Kennison, K.A. Ketchum, B.E. Kimmel, C.D. Kodira, C. Kraft, S. Kravitz, D. Kulp, Z. Lai, P. Lasko, Y. Lei, A.A. Levitsky, J. Li, Z. Li, Y. Liang, X. Lin, X. Liu, B. Mattei, T.C. McIntosh, M.P. McLeod, D. McPherson, G. Merkulov, N.V. Milshina, C. Mobarry, J. Morris, A. Moshrefi, S.M. Mount, M. Moy, B. Murphy, L. Murphy, D.M. Muzny, D.L. Nelson, D.R. Nelson, K.A. Nelson, K. Nixon, D.R. Nusskern, J.M. Pacleb, M. Palazzolo, G.S. Pittman, S. Pan, J. Pollard, V. Puri, M.G. Reese, K. Reinert, K. Remington, R.D.C. Saunders, F. Scheeler, H. Shen, B. C. Shue, I. Sidén-Kiamos, M. Simpson, M.P. Skupski, T. Smith, E. Spier, A.C. Spradling, M. Stapleton, R. Strong, E. Sun, R. Svirskas, C. Tector, R. Turner, E. Venter, A.H. Wang, X. Wang, Z.-Y. Wang, D.A. Wassarman, G.M. Weinstock, J. Weissenbach, S.M. Williams, T. Woodage, K.C. Worley, D. Wu, S. Yang, Q. Alison Yao, J. Ye, R.-F. Yeh, J.S. Zaveri, M. Zhan, G. Zhang, Q. Zhao, L. Zheng, X.H. Zheng, F.N. Zhong, W. Zhong, X. Zhou, S. Zhu, X. Zhu, H.O. Smith, R.A. Gibbs, E.W. Myers, G.M. Rubin, and J.C. Venter. 2000. The Genome Sequence of *Drosophila melanogaster*. *Science* **287(5461):** 2185-2195.

Alizadeh, F., Karp, R.M., Weisser, D.K., and G. Zweig. 1995. Physical mapping of chromosomes using unique probes. *Algorithmica* **13(1/2):** 52-76.

Allard, W. J., I. S. Sigal, and R. A. F. Dixon. 1987. Sequence of the gene encoding the human M1 muscarinic acetylcholine receptor. *Nucleic Acids Research* **15(24):** 10604.

Altschul, S.F., Miller, W., Myers, E.W., and D.J. Lipman. 1990. Basic Local Alignment Search Tool. *Journal of Molecular Biology* **215:** 403-410.

Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., and D.J. Lipman. 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research* **25:** 3389-3402.

Antequera, R. and A. Bird. 1993. Number of CpG islands and genes in human and mouse. *Genetics* **90:** 11995-11999.

Arratia, R., Lander E. S., Tavare S. and M.S. Waterman. 1991. Genomic mapping by anchoring random clones: a mathematical analysis. *Genomics* **11:** 806-827.

Batzer, M.A., Deininger, P.L., Hellmann Blumberg, U., Jurka, J., Labuda, D., Rubin, C.M., Schmid, C.W., Zietkiewicz, E., and Zuckerkandl, E. 1996. Standardized nomenclature for Alu repeats. *Journal of Molecular Evolution* **42:** 3-6.

Bassett, D. E., M. S. Boguski, F. Spencer, R. Reeves, S. H. Kim, T. Weaver, and P. Hieter. 1997. Genome Cross-Referencing and XREFDB – Implications for the identification and analysis of genes mutated in Human Disease. *Nature Genetics* **15(4):** 339-344.

Batzoglou, S., Berger, B., Kleitman, D.J., Lander, E.S., and L. Pachter. Recent developments in computational gene recognition. 1998. *Documenta Mathematica, Extra Volume ICM 1998* **1:** 649-658.

Batzoglou, S., Berger, B., Mesirov, J., and E.S. Lander. 1999. Sequencing a Genome by Walking with Clone-end Sequences: A Mathematical Analysis. *Genome Research* **9(12):** 1163-1174. Abstract in *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, RECOMB 2000* p.45.

Batzoglou, S. and S. Istrail. 1999. Physical Mapping with Repeated Probes: The Hypergraph Superstring Problem. *Lecture Notes in Computer Science, vol. 164. Special issue on CPM '99.*

Batzoglou, S., Pachter, L., Mesirov, J.P., Berger, B., and E.S. Lander. 2000. Human and Mouse Gene Structure: Comparative Analysis and Application to Exon Prediction. *Genome Research, In Press.* Abstract in *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, RECOMB 2000* p.46-53.

Benson, G. 1997. Sequence alignment with tandem duplication. *Journal of Computational Biology* **4(3):** 351-367.

Boguski, M. S., D. R. Cox, and R. M. Myers. 1996. Genomes and evolution – Overview. *Current Opinion in Genetics & Development* **6(6):** 683-685.

Booth K. S. and Lueker G. S. 1976. Testing for the consecutive ones property, interval graphs and planarity using PQ-tree algorithms. *Journal of Computer Systems Science* **13:** 335-379.

Burge, C. 1997. Identification of genes in human genomic DNA. *Ph.D. dissertation, Stanford University, Department of Mathematics.*

Burge, C. and S. Karlin. 1997. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Bioogy* **268:** 78-94.

Burset, M. and R. Guigo. 1996. Evaluation of gene structure prediction programs. *Genomics* **34(3):** 353-367.

Cech, T.R. 1993. Catalytic RNA: structure and mechanism. Biochemical Society Transaction **21:** 229-234.

The C. elegans Sequencing Consortium. 1998. Genome sequence of the nematode C. elegans: a platform for investigating biology. *Science* **282:** 2012-2018.

Celeste, A. J., V. Rosen, J. L. Buecker, R. Kriz, E. A. Wang, and J. M. Wozney. 1986. Isolation of the human gene for bone gla protein utilizing mouse and rat cDNA clones. *The EMBO Journal.* **5(8):** 1885-1890.

Collins, F., and D. Galas. 1993. A new five-year plan for the U.S. Human Genome Project. *Science* **262(5):** 43-46.

Collins, J.E., Cole, C.G. , Smink, L .J. , Garrett, C.L., Leversha, M.A., Soderlund, C.A., Maslen, G.L., Everett, L.A., Rice, C.M., Coffey, A.J., Gregory, S. G., Gwilliam, R., Dunham, A., Davies, A.F., Hassock, S., Todd, C.M., Lehrach, H, Hulsebos, T.J.M., Weissenbach, J., Morrow, B., Kucherlapati, R.S., Wadey. R., Scambler, P.J., Kim, U-J., Simon, M.I., Carter, N.P., Durbin, R, Dumanski, J.P., Bentley, D.R., and I. Dunham. 1995. A High Resolution Integrated Yeast Artificial Chromosome Clone Map of Human Chromosome 22. *Nature* **377:** 367-379.

Collins, F.S., Patrinos, A., Jordan, E., Chakravarti, A., Gesteland, R., Walters, L., and the members of the DOE and NIH planning groups. 1998. New goals for the U.S. Human Genome Project. *Science* **282(23):** 682-689.

Cormen, T., Leiserson, C.E., Rivest, R.L. 1990. Introduction to Algorithms. *MIT Press.*

Coulson, A., Sulston, J., Brenner, S., and J. Karn. 1986. Towards a physical map of the genome of the nematode Caenorhabditis elegans. *Proceedings of the National Academy of Science.* **83:** 7821-7825.

Dayhoff, M., R. M. Schwartz, and B. C. Orcutt. 1978. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure* **5:** 345-352.

Delcher, A.L., Kasif, S., Fleischmann, R.D., Peterson, J., White, O., and S.L. Salzberg. 1999. Alignment of whole genomes. *Nucleic Acids Research* **27(11):** 2369-2376.

Delphin, M. E., P. A. Stockwell, W. P. Tate, and C. M. Brown. 1999. Transterm, the translational signal database, extended to include full coding sequence and untranslated regions. *Nucleic Acids Research* **27:** 293-294.

Dujon, B. 1996. The yeast genome project: what did we learn? *Trends in Genetics* **12:** 263-270.

Edwards, A., and C.T. Caskey. Closure strategies for random DNA sequencing. 1991. *Methods: a Companion to Methods Enzymology* **3:** 41-47, Academic Press, New York.

Erdos, P. 1993. *Personal communication.*

Ewing, B. and P. Green. 1998. *Genome Research* **8:** 186-194.

Fasulo, D., Jiang, T., Karp, R.M., Settergren, R.J., and E. Thayer. 1999. Algorithmic Approach to Multiple Complete Digest Mapping. *Journal of Computational Biology* **6(2)**.

Fishburn, P. 1985. Interval orders and interval graphs. *Wiley, New York.*

Fleischmann, R.D., Adams, M.D., White, O., Clayton, R.A., Kirkness, E.F., Kerlavage, A.R., Bult, C.J., Tomb, J.F., Dougherty, B.A., Merrick, J.M., McKenny, K., Sutton, G., FitzHugh, W., Fields, C., Gocayne, J.D., Scott, J., Shirley, R., Liu, L.I., Glodek, A., Kelly, J.M., Weidman, J.F., Phillips, C.A., Spriggs, T., Hedblom, E., Cotton, M.D., Utterback, T.R., Hanna, M.C., Nguyen, D.T., Saudek, D.M., Brandon, R.C., Fine, L.D., Fritchman, J.L., Fuhmann, J.L., Geoghagen, N.S.M., Gnehm, C.L., McDonald, L.A., Small, K.V., Fraser, C.M., Smith, H.O., and J.C. Venter. 1995. Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. *Science* **269:** 496-511.

Foote, S., Vollrath, D., Hilton, A., and D. Page. 1992. The human Y chromosome: Overlapping DNA clones spanning the euchromatic region. *Science* **258:** 60-66.

Gale, M.D. and K.M. Devos. 1998. Plant comparative genetics after 10 years. *Science* **282(23):** 656-659.

Gelfand, M. S., A. A. Mironov and P. A. Pevzner. 1996. Gene recognition via spliced sequence alignment. *Proceedings of the National Academy of Science* **93**: 9061-9066.

Ghosh, S.P. 1975.Consecutive storage of relevant records with high redundancy. *Communications of the ACM* **18:** 464-471.

Goebel, S.J., Johnson, G.P., Perkus, M.E., Davis, S.W., Winslow, J.P., and E. Paoletti. 1990. The complete DNA sequence of Vaccinia virus. *Virology* **179:** 247-266.

Green E. D. and P. Green. 1991. Sequence-tagged site (STS) content mapping of human chromosomes: theoretical considerations and early experiences. *PCR Methods and Applications* **1:** 78-90.

Green, P. 1997. Against a whole-genome shotgun. *Genome Res*earch **7:** 410-417.

Greenberg, D.S., and S. Istrail. 1995. Physical mapping by STS hybridization: algorithmic strategies and the challenge of software evaluation. *Journal of Computational Biology* **2(2):** 219-274.

Griffiths, A.J.F., Miller, J.H., Suzuki, D.T., Lewontin, R.C., and W.M. Gelbart. 1993. An Introduction to Genetic Analysis. *Fifth Edition, W.H.Freeman and Company, New York.*

Hardison, R. C., J. Oeltjen, and Webb Miller. 1997. Long Human-Mouse Sequence Alignments Reveal Novel Regulatory Elements: A Reason to Sequence the Mouse Genome. *Genome Research* **7:** 959-966.

Heusel, J. W., E. M. Scarpati, N. A. Enkins, D. J. Gilbert, N. G. Copeland, S. D. Shapiro, and T. J. Ley. 1993. Molecular cloning, chromosomal location, and tissue-specific expression of the murine cathepsin G gene. *Blood.* **81(6):** 1614-1623.

Hodgkin, J., Horvitz, H.R., Jasny, B.R., and J. Kimble. *C.elegans*: sequence to biology. 1998. *Science* **(11282):** 2011.

Hudson. T. J., L. D. Stein, S. S. Gerety, J. Ma, A. B. Castle, J. Silva, D. K. Slonim, R. Baptista, L. Kruglyak, S.H. Xu, X. Hu, A. M. E. Colbert, C. Rosenberg, M. P. Reeve-Daly, S. Rozen, L. Hui, X. Wu, C. Vestergaard, K. M. Wilson, J. S. Bae, S. Maitra, S. Ganiatsas, C. A. Evans, M. M. DeAngelis, and K. A. Ingalls. 1995. An STC-based map of the human genome. *Science* **270(5):** 1945-1954.

Hunt, A.R.; J. E. Collins; R. Bruskiewich; D. M. Beare; M. Clamp; L. J. Smink; R. Ainscough; J. P. Almeida; A. Babbage; C. Bagguley; J. Bailey; K. Barlow; K. N. Bates; O. Beasley; C. P. Bird; S. Blakey; A. M. Bridgeman; D. Buck; J. Burgess; W. D. Burrill; J. Burton; C. Carder; N. P. Carter; Y. Chen; G. Clark; S. M. Clegg; V. Cobley; C. G. Cole; R. E. Collier; R. E. Connor; D. Conroy; N. Corby; G. J. Coville; A. V. Cox; J. Davis; E. Dawson; P. D. Dhami; C. Dockree; S. J. Dodsworth; R. M. Durbin; A. Ellington; K. L. Evans; J. M. Fey; K. Fleming; L. French; A. A. Garner; J. G. R. Gilbert; M. E. Goward; D. Grafham; M. N. Griffiths; C. Hall; R. Hall; G. Hall-Tamlyn; R. W. Heathcott; S. Ho; S. Holmes; S. E. Hunt; M. C. Jones; J. Kershaw; A. Kimberley; A. King; G. K. Laird; C. F. Langford; M. A. Leversha; C. Lloyd; D. M. Lloyd; I. D. Martyn; M. Mashreghi-Mohammadi; L. Matthews; O. T. McCann; J. McClay; S. McLaren; A. A. McMurray; S. A. Milne; B. J. Mortimore; C. N. Odell; R. Pavitt; A. V. Pearce; D. Pearson; B. J. Phillimore; S. H. Phillips; R. W. Plumb;

H. Ramsay; Y. Ramsey; L. Rogers; M. T. Ross; C. E. Scott; H. K. Sehra; C. D. Skuce; S. Smalley; M. L. Smith; C. Soderlund; L. Spragon; C. A. Steward; J. E. Sulston; R. M. Swann; M. Vaudin; M. Wall; J. M. Wallis; M. N. Whiteley; D. Willey; L. Williams; S. Williams; H. Williamson; T. E. Wilmer; L. Wilming; C. L. Wright; T. Hubbard; D. R. Bentley; S. Beck; J. Rogers; N. Shimizu; S. Minoshima; K. Kawasaki; T. Sasaki; S. Asakawa; J. Kudoh; A. Shintani; K. Shibuya; Y. Yoshizaki; N. Aoki; S. Mitsuyama; B. A. Roe; F. Chen; L. Chu; J. Crabtree; S. Deschamps; A. Do; T. Do; A. Dorman; F. Fang; Y. Fu; P. Hu; A. Hua; S. Kenton; H. Lai; H. I. Lao; J. Lewis; S. Lewis; S.-P. Lin; P. Loh; E. Malaj; T. Nguyen; H. Pan; S. Phan; S. Qi; Y. Qian; L. Ray; Q. Ren; S. Shaull; D. Sloan; L. Song; Q. Wang; Y. Wang; Z. Wang; J. White; D. Willingham; H. Wu; Z. Yao; M. Zhan; G. Zhang; S. Chissoe; J. Murray; N. Miller; P. Minx; R. Fulton; D. Johnson; G. Bemis; D. Bentley; H. Bradshaw; S. Bourne; M. Cordes; Z. Du; L. Fulton; D. Goela; T. Graves; J. Hawkins; K. Hinds; K. Kemp; P. Latreille; D. Layman; P. Ozersky; T. Rohlfing; P. Scheet; C. Walker; A. Wamsley; P. Wohldmann; K. Pepin; J. Nelson; I. Korf; J. A. Bedell; L. Hillier; E. Mardis; R. Waterston; R. Wilson; B. S. Emanuel; T. Shaikh; H. Kurahashi; S. Saitta; M. L. Budarf; H. E. McDermid; A. Johnson; A. C. C. Wong; B. E. Morrow; L. Edelmann; U. J. Kim; H. Shizuya; M. I. Simon; J. P. Dumanski; M. Peyrard; D. Kedra; E. Seroussi; I. Fransson; I. Tapia; C. E. Bruder; K. P. O[@squo]Brien; I. Dunham. 1999. The DNA sequence of human chromosome 22. *Nature* **402(6761):** 489-495.

Hurowitz, E. 1999. http: //cmgm.stanford.edu/~hurowitz/chemotaxis/chemo1.html.

Jang, W., A. Hua, S. V. Spilson, W. Miller, B. A. Roe, and M. H. Meisler. 1999. Comparative Sequence of Human and Mouse BAC Clones from the mnd2 Region of Chromosome 2p13. *Genome Research* **9**: 53-61

Karp, R. M. 1993. Mapping the genome: some combinatorial problems arising in molecular biology. *SODA 1993* 278-285.

Klenk, H.P., Clayton, R.A., Tomb, J., White, O., Nelson, K.E., Ketchum, K.A., Dodson, R.J., Gwinn, M., Hickey, E.K., Peterson, J.D., Richardson, D.L., Kerlavage, A.R., Graham, D.E., Kyrpides, N.C., Fleischmann, R.D., Quackenbush, J., Lee, N.H., Sutton, G.G., Gill, S., Kirkness, E.F., Dougherty, B.A., McKenney, K., Adams, M.D., Loftus, B., Peterson, S., Reich, C.I., McNeil, L.K., Badger, J.H., Glodek, A., Zhou, L., Overbeek, R., Gocayne, J.D., Weidman, J.F., McDonald, L., Utterback, T., Cotton, M.D., Spriggs, T., Artiach, P., Kaine, B.P., Sykes, S.M., Sadow, P.W., D'Andrea, K.P., Bowman, C., Fujii, C., Garland, S.A., Mason, T.M., Olsen, G.J., Fraser, C.M., Smith, H.O., Woese, C.R. and J.C. Venter. 1997. The complete genome sequence of the hyperthermophilic, sulphate-reducing archaeon Archaeoglobus fulgidus. *Nature* **390(6658):** 364-370.

Koop, B. F. 1995. Human and rodent DNA sequence comparisons: a mosaic model of human evolution. *TIG* **11(9):** 369-379.

Koop, B. F. and L. Hood. 1994. Striking sequence similarity over almost 100 kilobases of human and mouse T-cell receptor DNA. *Nature Genetics* **7:** 48-53.

Kou, A.T. 1977. Polynomial complete consecutive information retrieval problems. *SIAM Journal of Computing* **6(1):** 67-75.

Krapivsky, P.L. 1992. Kinetics of Random Sequential Parking on a Line. *Journal of Statistical Physics* **69:** 135-150.

Kumar, A., A. Toscani, S. Rane, and E. P. Reddy. 1996. Structural organization and chromosomal mapping of JAK3 locus. *Oncogene* **13(9):** 2009-2014.

Lander, E. S. 1997. The new genomics- global views of biology. *Science* **274(5287)**: 536-539.

Lander, E.S. and M.S. Waterman. 1988. Genomic mapping by fingerprinting random clones: a mathematical analysis. *Genomics* **2:** 231-239.

Lander, E.S. and R.A. Weinberg. 2000. Genomics: journey to the center of biology. 2000. Science **287(5459):** 1777-1782.

Lamerdin, J. E., M. A. Montgomery, S. A. Stilwagen, L. K. Scheidecker, R. S. Tebbs, K. W. Brookman, L. H. Thompson, and A. V. Carrano. 1995. Genomic Sequence Comparison of the Human and Mouse XRCC1 DNA Repair Gene Regions. *Genomics* **25:** 547-554.

Leslie, N. D., E. B. Immerman, J. E. Flach, M. Florez, J. L. Fridovich-Keil, and L. Elsas. 1992. The human galactose-1 phosphate uridyltransferase gene. *Genomics* **14:** 474-480.

Lewin, B. 1996. Genes VI. *Oxford University Press.*

Lin, X, Samir Kaul; S. Rounsley; T.P. Shea; M.-I. Benito; C.D. Town; C.Y. Fujii; T. Mason; C.L. Bowman; M. Barnstead; T.V. Feldblyum; C.R. Buell; K.A. Ketchum; J. Lee; C.M. Ronning; H.L. Koo; K.S. Moffat; L.A. Cronin; M. Shen; G. Pai; S. Van Aken; L. Umayam; L.J. Tallon; J.E. Gill; M.D. Adams; A.J. Carrera; T.H. Creasy; H.M. Goodman; C.R. Somerville; G.P. Copenhaver; D. Preuss; W.C. Nierman; O. White; J.A. Eisen; S.L. Salzberg; C.M. Fraser; J.C. Venter. 1999. Sequence and analysis of chromosome 2 of the plant Arabidopsis thaliana. *Nature* **402(6763):** 761-768.

Lipski, W.Jr. 1976. Information storage and retrieval – mathematical foundations II. *Theoretical Computer Science* **3:** 183-212.

Lipski, W.Jr. 1978. On strings containing all subsets as substrings. *Discrete Mathematics* **21:** 253-259.

Lodish, H., Baltimore, D., Berk, A., Zipursky, S.L., Matsudaira, P., and J. Darnell. 1998. Molecular Cell Biology. Fifth Edidion. *Scientific American Books, New York.*

Lukashin A. V. and M. Borodovsky. 1998. GENEMARK.HMM: new solutions for gene finding. *Nucleic Acids Research* **26(4)**: 1107-1115.

Makalowski, W. and M. S. Boguski. 1998a. Synonymous and Nonsynonymous Substitution Distances are Correlated in Mouse and Rat Genes. *Journal of Molecular Evolution* **47(2):** 119-121.

Makalowski, W. and M. S. Boguski. 1998b. Evolutionary parameters of the transcribed mammalian genome: An analysis of 2,820 orthologous rodent and human sequences. *Proceedings of the National Academy of Science* **95:** 9407-9412.

Makalowski, W., J. Zhang, and M. S. Boguski. 1996. Comparative analysis of 1196 orthologous mouse and human full-length mRNA and protein sequences. *Genome Research* **6:** 8456-857.

Marshall Graves, J.A. 1998. *ILAR Journal* **39:** 48.

Maxam, A.M., and W. Gilbert. 1977. A new method for sequencing DNA. *Proceedings of the National Academy of Science* **74:** 560-564.

McCullough, A.J., and S.M. Berget. 1997. G triplets located throughout a class of small vertebrate introns enforce intron borders and regulate splice site selection. *Molecular and Cellular Biology* **17(8):** 4562-4571.

Meinke, D.W., Cherry, M.J., Dean, C., Rounsley, S.D., and M. Koornneef. 1998. Arabidopsis thaliana: a model plant for genome analysis. *Science* **282(5389):** 662-882.

Myers, E.W. 1999. Whole-Genome DNA Sequencing. *Computing in Science and Engineering* **1(3):** 33-43.

Myers, E.W., G.G. Sutton, A.L. Delcher, I.M. Dew, D.P. Fasulo, M.J. Flanigan, S.A. Kravitz, C.M. Mobarry, K.H.J. Reinert, K.A. Remington, E.L. Anson, R.A. Bolanos, H.-H. Chou, C.M. Jordan, A.L. Halpern, S. Lonardi, E.M. Beasley, R.C. Brandon, L. Chen, P.J. Dunn, Z. Lai, Y. Liang, D.R. Nusskern, M. Zhan, Q. Zhang, X. Zheng, G.M. Rubin, M.D. Adams, and J.C. Venter. 2000. A Whole-Genome Assembly of *Drosophila. Science* **287(5461):** 2196-2204.

Nagata, S., M. Tsuchiya, S. Asano, O. Yamamoto, Y. Hirata, N. Kubota, M. Oheda, H. Nomura, and T. Yamazaki. 1986. The chromosomal gene structure and tho mRNAs for human granulocyte colony-stimulating factor. *The EMBO Journal* **5(3):** 575-581.

Needleman, S.B. and C.D. Wunch. 1970. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology* **48:**443-453.

Nelson D.O. and T.P. Speed. 1994. Statistical issues in constructing high-resolution physical maps. *Statistical Science* **9(3):** 334-354.

O'Brien, S.J., Menotti-Raymond, M., Murphy, W.J., Nash, W.G., Wienberg, J., Stanyon, R., Copeland, N.G., Jenkins, N.A., Womack, J.E., and J.A. Marshall Graves. 1999. The promise of comparative genomics in mammals. *Science* **286(5439):** 458-481.

O'Brien, S.J., Seuanez, H.N., and J.C. Womack. 1988. *Annual Reviews of Genetics* **22:** 323.

Oda, K., Katsuyuki, Y., Ohta, E., Nakamura, Y., Takemura, M., Nozato, N., Akashi, K., Kanegae, T., Ogura, Y., Kohchi, T., and K. Ohyama. 1992. Gene organization deduced from the complete sequence of Liverwort Marchantia polymorpha mitochondrial DNA. *Journal of Molecular Biology* **223:** 1-7.

Oeltjen, J. C., T. Malley, D. M. Muzny, W. Miller, R. A. Gibbs, and J. W. Belmont. 1997. Large Scale Comparative Sequence Analysis of the Human and Murine Bruton's Tyrosine Kinase Loci Reveals Conserved Regulatory Domains. *Genome Research* **7:** 315-329.

Ohyama, K., Fukuzawa, H., Kohchi, T., Shirai, H., Sano, T., Sano, S., Umesono, K., Shiki, Y., Takeuchi, M., Chang, Z., Aota, S., Inokuchi, H., and H. Ozeki. 1986. Chloroplast gene organization deduced from complete sequence of livewort Marchantia polymorpha chloroplast DNA. *Nature* **322:** 572-574.

Oliver, S.G., *et al.* 1992. The complete DNA sequence of yeast chromosome III. *Nature* **357:** 38-46.

Olson, M.V., Dutchik, J.E., Graham, M.Y., Brodeur, G.M., Helms, C., Frank, M., MacCollin, M., Scheinman, R., and T. Frank. 1986. Random-clone strategy for genomic restriction mapping in yeast. *Proceedings of the National Academy of Sci*ence **83:** 7826-7830.

Pachter, L.S. 1999. Domino tiling, gene recognition, and mice. *Ph.D. dissertation MIT Department of Mathematics.*

Pachter, L., Batzoglou, S., Spitkovsky, V.I., Beebee, W., Lander, E.S., Kleitman, D.J., and B. Berger 1999. A dictionary based approach for gene annotation. *Journal of Computational Biology* **(6) 3-4:** 419-430. Abstract in *Proceedings of the Third Annual International Conference in Computational Molecular Biology, RECOMB '99.*

Papadimitriou, C.H. 1994. Computational Complexity. *Addison-Wesley Publishing Company.*

Papadimitriou, C.H. and M. Yannakakis. 1993. The traveling salesman problem with distances one and two. *Mathematics of Operations Research* **18:** 1-11.

Phizicky, E.M., and C.L. Greer. 1993. Pre-tRNA splicing: variation on a theme or exception to the rule? *Trends in Biochemical Science* **18:** 31-34.

Riedy, M. C., A. S. Dutra, T. B. Blake, W. Modi, B. L. Lal, J. Davis, A. Bosse, J. J. O'Shea, and J. A. Johnston. 1996. Genomic sequence, organization, and chromosomal localization of human JAK3. *Genomics* **37:** 57-61.

Roach, J. 1995. Random subcloning. *Genome Research* **5:** 464-473.

Roest Crollius, H., Jaillon, O., Dasilva, D., Bouneau, L., Fizames, C., Billault, A., Bernot, A., Quetier, F., Weissenbach, J., and W. Saurin. 1999. Exon Detection by Comparison Between Two Distant Vertebrate Genome Sequences. *The Second Georgia Tech International Conference in Bioinformatics, Poster.*

Ross, S.M. 1970. "Applied probability models with optimization applications". Holden-Day, San Fransisco.

Rubin, G.M. 1996. The Drosophila Genome Project. *Genome Research* **6:** 71-79.

Rubin, G.M., M.D. Yandell, J.R. Wortman, G.L. G. Miklos, C.R. Nelson, I.K. Hariharan, M.E. Fortini, P.W. Li, R. Apweiler, W. Fleischmann, J.M. Cherry, S. Henikoff, M.P. Skupski, S. Misra, M. Ashburner, E. Birney, M.S. Boguski, T. Brody, P. Brokstein, S.E. Celniker, S.A. Chervitz, D. Coates, A. Cravchik, A. Gabrielian, R.F. Galle, W.M. Gelbart, R.A. George, L.S.B. Goldstein, F. Gong, P. Guan, N.L. Harris, B.A. Hay, R.A. Hoskins, J. Li, Z. Li, R.O. Hynes, S.J.M. Jones, P.M. Kuehl, B. Lemaitre, J.T. Littleton, D.K. Morrison, C. Mungall, P.H. O'Farrell, O.K. Pickeral, C. Shue, L.B. Vosshall, J. Zhang, Q. Zhao, X.H. Zheng, F. Zhong, W. Zhong, R. Gibbs, J.C. Venter, M.D. Adams, and S. Lewis. 2000. Comparative Genomics of the Eukaryotes. *Science* **287(5461):** 2204-2215.

Sanger, F., Coulson, A.R., Barrell, B.G., Smith, A.J.H., and B.A. Roe. 1980. Cloning in single-stranded bacteriophage as an aid to rapid DNA sequencing. *Journal of Molecular Biology* **143:** 161-178.

Sanger, F., Coulson, A.R., Hong, G.F., Hill, D.F., and G.B. Petersen. 1982. Nucleotide sequence of bacteriophage λ DNA. *Journal of Molecular Biology* **162:** 729-773.

Sanger, F., Nickolen, S., and A.R. Coulson. 1977. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Science* **74:** 5463-5467.

Shamir, R. 1997. *Personal communication.*

Sharp, P.A. and C.B. Burge. 1997. Classification of introns. U2-type or U12-type. *Cell* **91:** 875-879.

Shehee, W. R., L. Loeb, N. B. Adey, F. H. Burton N. C. Casavant, P. Cole, C. J. Davies, R. A. McGraw, S. A. Schichman, D. M. Severynse, C. F. Voliva, F. W. Weyter, G. B. Wisely, M. H. Edgell, and C. A. Hutchison. 1989. Nucleotide Sequence of the BALB/c Mouse β-Globin Complex. *Journal of Molecular Biology* **205:** 41-62.

Sherman, F. 1997. Yeast genetics. *The Encyclopedia of Molecular Biology and Molecular Medicine* **6:** 302-325. Edited by R. A. Meyers, VCH Pub. Weinheim, Germany, 1997. Also in http: //www.urmc.rochester.edu/smd/biochem/yeast/.

Schmid, C. W. 1996. Alu: structure, origin, evolution, significance, and function of one-tenth of human DNA. *Progress Nucleic Acids Research Molecular Biology* **53:** 283-319.

Schuller, M.K., C., Wambutt, R., Murphy, G., Volckaert, G., Pohl, T., Dusterhoft A, Stiekema, W., Entian, K.D., Terryn, N., Harris, B., Ansorge, W., Brandt, P., Grivell, L., Rieger, M., Weichselgartner, M., de Simone, V., Obermaier, B., Mache, R., Muller, M., Kreis, M., Delseny, M., Puigdomenech, P., Watson, M., McCombie, W.R., et al. 1999. *Nature* **402(6763):** 769-77.

Siegel, A.F., Trask, B., Roach, J., Mahairas, G.G., Hood, L. and van der Engh, G. 1998. Analysis of sequence-tagged-connector strategies for DNA sequencing. *Genome Res*earch **9:** 297-307.

Smit, A.F.A. 1996. Origin of interspersed repeats in the human genome. *Current Opinion Genetic Development* **6(6):** 743-749.

Smit, A.F.A. 1995. Origin and evolution of mammalian interspersed repeats. *Ph.D. dissertation, USC.*

Smit, A.F.A. and A.D. Riggs. 1995. MIRs are classic, tRNA-derived SINEs that amplified before the mammalian radiation. *Nucleic Acids Research* **23:** 98-102.

Smith, T.F. and M.S. Waterman. 1981. Identification of common molecular subsequences. *Journal of Molecular Biology* **147:** 195-197.

Stryer, L. 1996. Biochemistry. *Fourth Edition, W.H. Freeman and Company, New York.*

Venter, J.C., Smith, H.O., and Hood, L. 1996. A new strategy for genome sequencing. *Nature* **381:** 364-366.

Watson, J.D. and F.H.C. Crick. 1953a. Molecular structure of nucleic acid. A structure for deoxyribose nucleic acid. *Nature* **171:** 737-738.

Watson, J.D. and F.H.C. Crick. 1953b. Genetic implications of the structure of deoxyribonucleic acid. *Nature* **171:** 964-967.

Weber, J.L. and E.W. Myers. 1997. Human whole-genome shotgun sequencing. *Genome Research* **7:** 401-409.

Yoneda, K., D. Hohl, O. W. McBride, M. Wang, K. U. Cehrs, W. W. Idler, and p. M. Steinert. 1992. The human loricrin gene. *The Journal of Biological Chemistry* **267(25):** 18060-18066.

Zhang, J. and T. L. Madden. 1997. PowerBLAST: A New Network BLAST Application for Interactive or Automated Sequence Analysis and Annotation. *Genome Research* **7:** 649-656.

http://theory.lcs.mit.edu/crosspecies/. 2000. *GLASS* and *ROSETTA* webpage.

http://Ag.Arizona.Edu/tree/. 1999. Tree of Life webpage.

http://www.*PHRAP*.com. 1999. *PHRAP* system webpage.

http://www.sanger.ac.uk/Software/Wise2/. 1999. *Wise2 webpage.*

http://www.ncbi.nlm.nih.gov/. 1998. National Center for Biology Information webpage.

http://www.ncbi.nlm.nih.gov/BLAST/. 1999. *BLAST* program family webpage.

http://ftp.genome.washington.edu/RM/RepeatMasker.html. 1999. *RepeatMasker* webpage.