

## ABSTRACT

Title of Thesis:           A Dynamical Systems Approach to  
                                  Estimating the Sequences of  
                                  Repeat Regions in the Genome

Degree candidate:        Kathleen Ann Meloney

Degree and year:         Master of Arts, 2004

Thesis directed by:     Dr. James A. Yorke  
                                  Department of Mathematics and Physics

In 1982, Fred Sanger [9] introduced a cloning technique on which shotgun sequencing is based. Shotgun sequencing is a method for determining the sequence of bases (or letters) in the genome and since its introduction, many groups have used this technique to sequence the genomes of various organisms. The shotgun technique involves breaking the DNA into a large number of small pieces, each of whose sequence of letters is determined experimentally. Current technology limits the length of the sequenced pieces to approximately 500 letters. Then, like a puzzle, the pieces are assembled using computer algorithms to produce the complete sequence. The greatest difficulty with the shotgun technique is the presence of subsequences longer than 500 letters that occur multiple times in the genome with minor variations.

We present a dynamical systems approach to estimating the sequence of letters of these long, highly repetitive subsequences in the genome. Our results suggest that this approach produces good representatives of the long repetitive subsequences in a genome. We also present potential applications of this method to genome assembly.

A Dynamical Systems Approach to  
Estimating the Sequences of  
Repeat Regions in the Genome

by

Kathleen Ann Meloney

Thesis submitted to the Faculty of the Graduate School of the  
University of Maryland, College Park in partial fulfillment  
of the requirements for the degree of  
Master of Arts  
2004

Advisory Committee:

Dr. James A. Yorke, Chairman/Advisor  
Dr. Brian Hunt  
Dr. Daniel Rudolph



# TABLE OF CONTENTS

<b>List of Tables</b>	<b>iv</b>
<b>List of Figures</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Sequencing a Genome . . . . .	2
1.1.1 Obtaining Reads . . . . .	3
1.1.2 Joining Reads and Creating Contigs . . . . .	4
1.2 Sources of Errors . . . . .	5
<b>2 The Deterministic Path-Building Algorithm</b>	<b>7</b>
2.1 Building the Graph . . . . .	7
2.2 Building Paths . . . . .	10
2.2.1 Classifying Paths . . . . .	10
<b>3 Our Data Set: Genomes Investigated</b>	<b>12</b>
<b>4 Results</b>	<b>14</b>
4.1 Characteristics of the Path-Builder . . . . .	14
4.1.1 Making Sound Choices . . . . .	14
4.1.2 Distribution of Path Types Across the Genomes . . . . .	16

4.2	Matching Paths to Finished Sequence . . . . .	17
4.2.1	The Rat . . . . .	18
4.2.2	The Fruit Fly . . . . .	18
4.2.3	The Chicken . . . . .	19
4.3	An Example of a Successful Path in the Fruit Fly . . . . .	21
4.4	Distinguishing Between Copies of a Repeat Region . . . . .	22
4.4.1	The Theory . . . . .	22
4.4.2	Applying the Theory to Our Data Sets . . . . .	24
<b>5</b>	<b>Conclusions and Future Work</b>	<b>28</b>
5.1	Potential Applications to Assembly . . . . .	29
5.1.1	Validation of Paths . . . . .	30
5.1.2	Avoiding Misassemblies due to Repeat Regions . . . . .	30
5.2	So Where from Here? . . . . .	31
5.3	Final Comments . . . . .	32
	<b>Bibliography</b>	<b>33</b>

## LIST OF TABLES

1.1	The table shows 600 copies of the same repetitive subsequence and the number of copies of each variant in a hypothetical genome. . .	2
2.1	This table contains twelve hypothetical reads. . . . .	7
2.2	This table lists all eight $(2 + 1)$ -mers with their frequencies found in the reads in Table 2.1. . . . .	8
4.1	This table gives the percent of nodes (representing 20-mers) in each genome's graph having deterministic values of at least 100%, 90%, 80%, 70%, 60%, and 50%. There were 720064, 651950, and 19719 20-mers in the library of the chicken, fly, and rat, respectively. The rat BAC gsfk was selected for this table because it had the worst determinsitic values of all of the 14 BACs. . . . .	16
4.2	The table above gives the percent of 20-mers in the graph of each genome that belong to a periodic, eventually periodic, or terminating path. . . . .	17
4.3	The table above highlights some interesting features of the set of paths built using 20-mers from the reads for each of the genomes.	20

4.4	The table gives the percent of nodes in each graph that have no outgoing edges. Notice that the chicken has the greatest percent of nodes with no outgoing edges. This is mainly due to the proliferation of <i>N</i> 's in the reads for the chicken.(An <i>N</i> is a low-confidence letter.) . . . . .	20
4.5	The above table highlights 20 disjoint subsequences of the finished sequence that match an artificially created path of length 6839 letters for the genome of the fruit fly. These subsequences were selected because they each match the path for more than 1000 letters with a percent error less than 1%. The first column is the name of the chromosome containing the matching subsequence. The second column gives the length of the subsequence that matches the path. The third, fourth, and fifth columns outline the mismatches between the subsequence and the path. Insertions, I, are letters occurring in the finished sequence, but not in the path; deletions, D, are letters occurring in the path, but not found in the finished sequence; substitutions, S, are letters that disagree between the path and the subsequence. The percent error is the sum of columns 2, 3 and 4 divided by the length of the match. Notice that there is one perfect match of length 4138 in chromosome 2L. . . . .	26
4.6	The table above contains the values for $r_{ij}$ for the rat path and its thirteen matches. . . . .	27



## LIST OF FIGURES

1.1	First, the DNA is shattered into inserts. Then the ends of the inserts are sequenced to produce the two reads $X$ and $Y$ . . . . .	4
1.2	Since reads $X$ and $Y$ overlap one another, they are glued together to form the longer strand $Z$ . Other overlapping reads are glued together to form a unitig. . . . .	5
1.3	Misassembly due to a repetitive sequence. Consider the hypothetical sequence of DNA containing two nearly identical copies of a repeat $R'$ and $R''$ and three stretches of unique sequence, $A$ , $B$ , and $C$ . One read, $Z$ , straddles $R'$ and $B$ , and a second read, $W$ , straddles $R''$ and $C$ . Because $R'$ and $R''$ are nearly identical, $Y$ will overlap with both $W$ and $Z$ , but $Z$ and $W$ will disagree beyond the repeat section of the reads. It is unclear as to whether $Z$ or $W$ should be joined with $Y$ . If $W$ is chosen, then the assembly incorrectly moves from $A$ through one copy of the repeat region to $C$ , skipping $B$ . . . . .	6

2.1	The graph on the left contains all occurring $k$ -mers and all occurring edges for the database in Table 2.2. Nodes for which the total of the outgoing edge labels is less than the cutoff of 2 have been marked with a red $\times$ and are removed in the middle graph. In the center graph, $TG$ has two outgoing edges. Since the label on the edge $TGC$ is the smaller of the two, it is marked with a red $\times$ to denote that it will be removed. The final graph is shown on the right. . . . .	9
2.2	In the figure above, the node AA is labeled with a “P” to denote that it belongs to a periodic cycle. The node CA is labeled with a “T” because it has no outgoing edges. . . . .	11
4.1	The path $P$ is shown in the middle row. Mismatches between any of the four matching regions, $M_1, \dots, M_4$ , and the path are shown as $\times$ 's and labeled with the base. Regions of agreement are not marked. Aligning bases are highlighted with vertical dotted lines.	23
4.2	The figure above illustrates the groups that are created when we group those matches $M_i$ and $M_j$ for which $r_{ij} < .4$ . Notice the interesting interaction between $M_6, M_{13}, M_{11}, M_3$ , and $M_4$ . . . . .	25
5.1	Overlapping reads $M_1, M_2, \dots, M_n$ are first compared against the path $P$ , then they are pairwise compared. Finally, $r_{ij}$ is calculated for each pair and this value is used to group the reads according to the copy of the repeat $R$ with which they belong. . . . .	31

# Chapter 1

## Introduction

Inside the cells of every living organism is a chemical “instruction manual” which contains all the necessary information to construct the entire organism. This manual, called **Deoxyribonucleic Acid (DNA)** is stored as a long sequence of four molecules: adenine (A), guanine (G), cytosine (C), and thymine (T), called nucleotide bases, whose order encodes the structure of every part of the organism. These same four bases can encode the directions to create a fish’s tail, a bird’s wing, an elephant’s trunk, or a human’s eye depending on their sequence in the strand of DNA. This sequence is called the **genome**.

Each strand of DNA contains many genes, which are the basic physical and functional units of heredity. A **gene** is a specific sequence of bases in the DNA that codes for proteins, the building blocks of tissues and organs. Genes comprise approximately 10% of the human genome [2]. The remaining 90% of the human genome does not code for proteins, but includes directions and markers to guide the production of proteins in a given cell. These non-coding regions of a genome contain a large number of repetitive subsequences. These are subsequences of significant length that appear multiple times in a genome

sequence	number of copies
ATTAGGCATGTGCCA	300
TTTAGGCATGTGCCA	100
ATTAGGCATTTGCC	200

Table 1.1: The table shows 600 copies of the same repetitive subsequence and the number of copies of each variant in a hypothetical genome.

(more than would occur by random chance). A repeat region may be a specific sequence, such as *ACGTAGCTAGGGTCAGTCAGGGTCAGGTAGAGA*, that appears in several different locations. Some of these regions are **tandem repeats**—several copies of short subsequences of DNA placed side-by-side, such as the tandem repeat of *ACG* in the sequence *ACGACGACGACGACGACGACG*. When they appear in a genome, different copies of the same repeat sequence can have minor variations—bases that are different in the different copies or smaller subsequences that have been added or removed.

For example, consider the sequence shown in the first row of Table 1.1. Suppose that there are 300 copies of this sequence in a hypothetical genome. Often there are also many copies of closely related sequences; in the example in Table 1.1, 100 copies have one base that is different from the original sequence, and 200 copies that have two bases that are different.

## 1.1 Sequencing a Genome

In the past 20 years, there has been much work in the field of genomics. In particular, researchers have been very interested in finding the genomic sequences of different organisms. Finding the sequence of a genome is a difficult

and long process. The first draft of the chicken genome contains long stretches of *N*'s, which represent gaps (of unknown letters) in the sequence. Closing these gaps involves passing through the assembly process several more times. This often does not fill all of the gaps, which must be closed using time-consuming (and often expensive) laboratory techniques. Sequencing a genome is a difficult process that is constantly being studied in an attempt to improve the quality of the final sequence and the speed of the process.

### 1.1.1 Obtaining Reads

The process of sequencing begins by shattering the DNA into thousands of tiny segments called **inserts**. There are two relevant methods of obtaining inserts. In **whole-genome shotgun sequencing (WGSS)**, multiple copies of the entire genome are created in the lab. Then, the set of copies are shattered into a large number of inserts. In **bacterial artificial chromosome** sequencing, the DNA is first broken into a set of larger segments called bacterial artificial chromosomes (BACs). Then the BACs are mapped to the genome to create a tiling. Finally, each BAC is shattered into a large number of inserts (separately by BAC). In either method, the sequence of bases in each insert is determined using a series of chemical reactions. However, current technology limits the number of bases that can be sequenced at a time. Laboratory techniques can accurately sequence only about the first 500 bases on each end of an insert. Since the inserts tend to be longer than 1000 bases, this results in only the ends of each insert being sequenced. The sequenced ends of each insert are called **reads**. (See Figure 1.1.) Reads obtained by whole-genome shotgun sequencing are called WGS reads and those obtained using bacterial artificial chromosomes

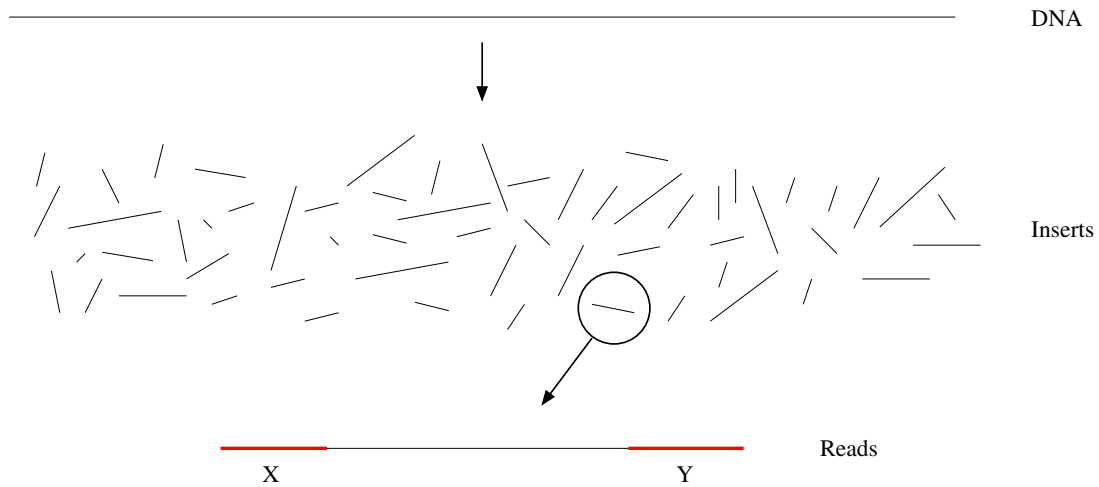


Figure 1.1: First, the DNA is shattered into inserts. Then the ends of the inserts are sequenced to produce the two reads  $X$  and  $Y$ .

are called BAC reads.

Occasionally, that some inserts will not be sequenced or will be inaccurately sequenced. To prevent this from causing a problem, enough inserts are created so that any given base in the genome will be contained in multiple reads. This is done in such a way that the reads are uniformly distributed. The number of reads covering a given base is a Poisson distribution and the mean of this distribution is called the **average coverage** of the genome. Since it is unknown from where each insert came, we are left with a 1-dimensional puzzle that we must assemble, but instead of interlocking, adjacent pieces overlap each other.

### 1.1.2 Joining Reads and Creating Contigs

Thus, the next step in the sequencing process is called **overlapping**. Those pairs of reads whose ends match with high fidelity are called **overlaps**. Suppose reads  $X$  and  $Y$  overlap one another as in Figure 1.2. These two overlapping

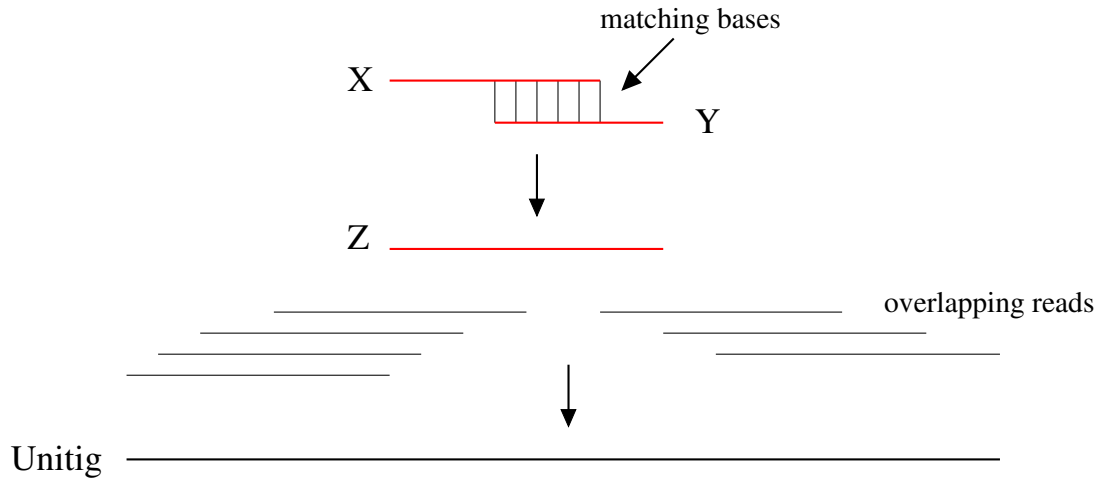


Figure 1.2: Since reads  $X$  and  $Y$  overlap one another, they are glued together to form the longer strand  $Z$ . Other overlapping reads are glued together to form a unitig.

reads are glued together to form one strand  $Z$ , and the ends of  $Z$  are then overlapped with new reads, and the process continues. In this way, larger pieces of the genome are reconstructed in a step-wise fashion.

## 1.2 Sources of Errors

Since the sequencing process is imperfect and the structure of the genome is complex, many problems can arise during sequencing. In particular, during the overlapping phase, it is possible that a read has multiple overlaps. If one of the overlaps is randomly chosen to continue the extension, it is possible that two regions of the genome that should not be placed next to each other are joined. This often happens in repeat regions that are longer than 500 bases, because several very similar copies of the same sequence occur in different areas of the genome, and an entire copy of the repeat region is not contained within a single read. Thus, reads from these copies will be nearly identical and will overlap one

another during the assembly stage. If the incorrect read is chosen to continue the extension, ( $W$  is the incorrect read in Figure 1.3) the unique region between the two copies of the repeat region will not be placed correctly, resulting in a misassembly. (See Figure 1.3.)

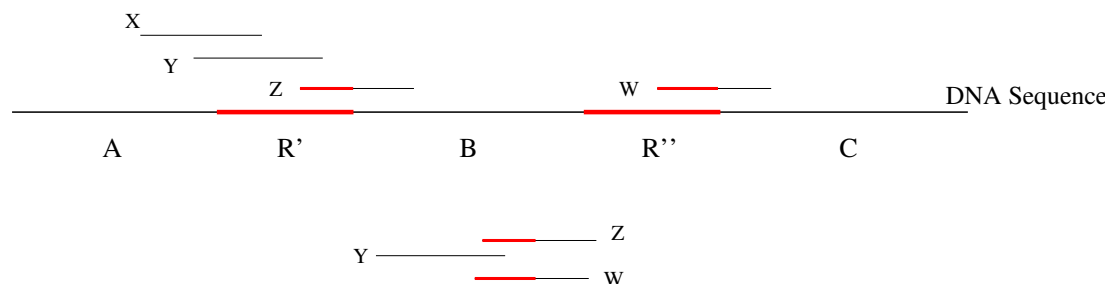


Figure 1.3: Misassembly due to a repetitive sequence. Consider the hypothetical sequence of DNA containing two nearly identical copies of a repeat  $R'$  and  $R''$  and three stretches of unique sequence,  $A$ ,  $B$ , and  $C$ . One read,  $Z$ , straddles  $R'$  and  $B$ , and a second read,  $W$ , straddles  $R''$  and  $C$ . Because  $R'$  and  $R''$  are nearly identical,  $Y$  will overlap with both  $W$  and  $Z$ , but  $Z$  and  $W$  will disagree beyond the repeat section of the reads. It is unclear as to whether  $Z$  or  $W$  should be joined with  $Y$ . If  $W$  is chosen, then the assembly incorrectly moves from  $A$  through one copy of the repeat region to  $C$ , skipping  $B$ .

Thus, misassemblies can occur when repeat regions are not detected during assembly. Correcting misassemblies is much more difficult after assembly is completed and incorrect assemblies result in incorrect genome sequences [5]. Several methods of detecting repeats have been developed, but repeats continue to confound assembly, even under the most stringent conditions.



## Chapter 2

### The Deterministic Path-Building Algorithm

We have created a deterministic algorithm that will make artificial subsequences of letters from the most highly repetitive elements of the genome with the goal of producing representative repeat regions. These subsequences, which we call **paths**, can potentially be used to identify copies of repeat regions.

#### 2.1 Building the Graph

The first step in building paths is the creation, for a given positive integer  $k$ , of a library of all subsequences of  $k + 1$  letters, called  $(k + 1)$ -mers, and their frequencies of occurrence in the set of reads for a genome. In practice, we use  $k \geq 20$ . For example, consider the set of twelve hypothetical reads shown in Table 2.1. The library of  $(2 + 1)$ -mers with their frequencies for this set of reads

TGAAAAAA	GAAAA	TGCAC	CGCA	TGCA	TCA
TGAAA	TGAAA	TCAC	TCAC	TCAC	TCA

Table 2.1: This table contains twelve hypothetical reads.

is in Table 2.2.

For each  $(k + 1)$ -mer in the library, we also add to its frequency the

(2 + 1)-mer frequency	AAA 8	CAC 4	CGC 1	GAA 4	GCA 3	TCA 5	TGA 3	TGC 2
--------------------------	----------	----------	----------	----------	----------	----------	----------	----------

Table 2.2: This table lists all eight  $(2 + 1)$ -mers with their frequencies found in the reads in Table 2.1.

frequency of its reverse complement. DNA is a double-stranded molecule. The reverse complement of a subsequence is the subsequence that is opposite it in the other strand of DNA; for example, the reverse complement of *GCA* is *TGC* since T is opposite A, C is opposite G, and the reverse complement is read backwards on the other strand of DNA.

Each  $(k + 1)$ -mer determines a transition between two successive overlapping  $k$ -mers by specifying that its first  $k$  letters precede its last  $k$  letters. For example the  $(2 + 1)$ -mer CAC specifies that the 2-mer CA may precede the 2-mer AC. Next, we build a directed graph whose nodes are  $k$ -mers and whose edges are  $(k + 1)$ -mers. We label each edge with the frequency the  $(k + 1)$ -mer. In the example above, the edge connecting CA to AC is labeled with a 4 because CAC occurs four times (see Figure 2.1).

At this point, each node will have at most four outgoing edges. The graph of 2-mers from the example above is given in Figure 2.1. For each node, the sum of the labels on its outgoing edges is total number of times the corresponding  $k$ -mer occurs as the first  $k$  letters of a  $(k + 1)$ -mer. (If a  $k$ -mer occurs at the end of a read, it is the last  $k$ -letters of a  $(k + 1)$ -mer and will have no outgoing edge from that particular occurrence.) We want to represent only the most highly repetitive elements in the genome; thus, we remove any nodes for which this total is less than some threshold number, called the **cutoff** number (in Figure 2.1, the cutoff is 2.). Since our library is built using the reads

and we want to limit our path-builder to only those  $k$ -mers that occur with a high frequency, we choose the cutoff number to reflect how often we would expect to see any  $k$ -mer in the genome in the set of reads. If we choose the cutoff number to be greater than  $2n$ , where  $n$  is the genome's average coverage by reads, then the nodes remaining in the graph correspond to  $k$ -mers that probably occur at least twice in the genome.

Finally, for each node, we remove all outgoing edges except the outgoing edge having the greatest label number. This is equivalent to picking the most frequent  $(k + 1)$ -mer out of the set of four  $(k + 1)$ -mers whose first  $k$  letters are the same. In the case of a tie, choose one; for example, choose the  $(k + 1)$ -mer whose last letter appears first in the alphabet. In practice, it does not often happen that two edges have the same or nearly the same label number. By eliminating the edges with smaller label numbers, we limit ourselves to the most frequently occurring  $(k + 1)$ -mers (transitions) in the genome.

We now have a directed graph in which each node can have at most four incoming edges and at most one outgoing edge. (See Figure 2.1.)

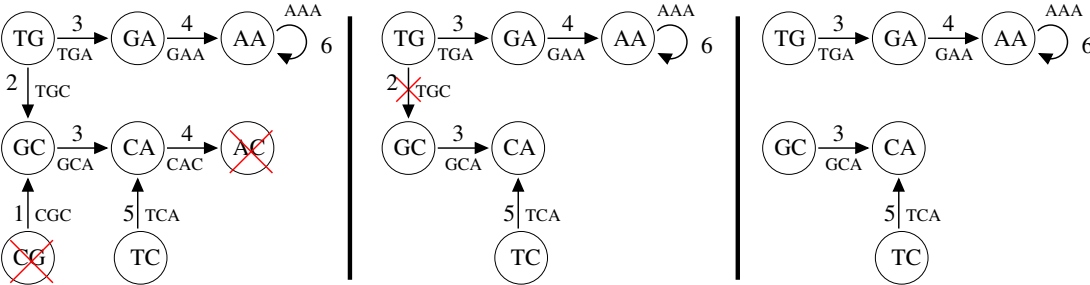


Figure 2.1: The graph on the left contains all occurring  $k$ -mers and all occurring edges for the database in Table 2.2. Nodes for which the total of the outgoing edge labels is less than the cutoff of 2 have been marked with a red  $\times$  and are removed in the middle graph. In the center graph,  $TG$  has two outgoing edges. Since the label on the edge  $TGC$  is the smaller of the two, it is marked with a red  $\times$  to denote that it will be removed. The final graph is shown on the right.

## 2.2 Building Paths

The path-builder creates a “forward” path through the graph by beginning at a node and following the directed edges through the graph until it encounters a node that it has already visited or until it encounters a node that has no outgoing edges. As it follows the path through the graph, it creates the sequence of letters determined by the  $k$ -mers corresponding to the nodes in the path.

For example, suppose we wanted to build a path through the graph in Figure 2.1 by starting at the node labeled  $TG$ . First, we would follow the edge to the node  $GA$ . Since the directed edge from the node  $TG$  to  $GA$  represents the 3-mer  $TGA$ , the path is now  $TGA$ . The next step is to  $AA$ , and thus the path is now  $TGAA$ . Finally, the third step returns to the node  $AA$ , so the path stops since it will forevermore cycle to the node  $AA$ . Thus, the path corresponds to the sequence  $TGAAAAAAAA\dots$ . In practice, we terminate a periodic loop after one cycle, so we would write  $AAAAAAAA\dots$  as  $AAA$ . In this manner, we create a path starting at each node in the graph.

### 2.2.1 Classifying Paths

Notice that some nodes in the graph are part of a periodic cycle and some nodes have no outgoing edges. We can label any node belonging to a periodic cycle with a “P” and any node with no outgoing edges “T” (to denote it as a terminating node). (See Figure 2.2.) Any path through the graph begins at some node, which is labeled by its corresponding  $k$ -mer. Since a path starting at any given node is completely determined (each node has at most one outgoing edge), we can classify a path as either **periodic**, **eventually periodic**, or **terminating** based on the labels (P or T) of the nodes in the

path. A path that begins on a “P” node is periodic. A path that begins on an unlabeled node and encounters a “P” node is classified as eventually periodic. A path that begins on an unlabeled node and encounters a “T” node is classified as terminating. Thus, for a given graph, there are at most two sets of sinks to which all paths are attracted: the set of nodes ( $k$ -mers) belonging to a periodic cycle and the set of terminating nodes.

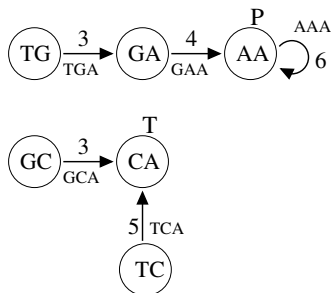


Figure 2.2: In the figure above, the node AA is labeled with a “P” to denote that it belongs to a periodic cycle. The node CA is labeled with a “T” because it has no outgoing edges.

It does not make sense to consider the path  $GAAA\dots$  as a different path from  $TGAAA\dots$  since  $GAAA\dots$  is simply a subsequence of  $TGAAA\dots$ . Thus, we retain only those paths that start at nodes having no incoming edges, unless it is a purely periodic path. In the example above, there are four paths:  $TGAAA\dots$ ,  $GCA$ ,  $TCA$ , and  $AAA\dots$ , where the last is retained since it is purely periodic.

## Chapter 3

### Our Data Set: Genomes Investigated

Although we can test our algorithm on a set of reads from any genome, we focus our attention on the genomes of three different organisms: *Rattus norvegicus* (rat), *Drosophila melanogaster* (fruit fly), and *Gallus gallus*, a type of chicken. Since these organisms have relatively long, complicated genomes, and both the reads and some parts of the finished sequences are available, the genomes of these animals make particularly good test cases.

During this project, the genome of *Rattus norvegicus* was being completed, so the majority of the 2.75 billion base pairs was sequenced. The genome was sequenced by a team headed by the Human Genome Sequencing Center at the Baylor College of Medicine using 7 times coverage by reads [4]. The set of reads we used were error-corrected using methods in [8] and were grouped according to the BACs<sup>1</sup> with which they belonged. (See [1] for a discussion of “binning.”) We look at 14 BACs.<sup>2</sup> Each BAC is approximately 200,000 letters long. These 14 BACs were selected because the reads and

---

<sup>1</sup>For a definition of BAC, see page 3.

<sup>2</sup>The BACs studied were those named gybz, gdwn, gexm, ggkp, gixt, gqqd, grmx, gsfk, gsta, gtgf, gxfc, gzjc, kbqm, and kdfe.

finished sequence were readily available. A cutoff number of 10 (about  $1.5n$ , where  $n$  is the average coverage of the genome by reads) was used in constructing the graph for each BAC. (See Section 2.1 for a discussion of cutoff numbers.)

The first draft of the genome of *Drosophila melanogaster* was released in 2000 by Celera Genomics, who used a whole-genome shotgun sequencing approach [4]. Since then, several improved versions of the genome have been released. The genome has approximately 120 million base pairs and contains approximately 14,000 genes. It was sequenced using WGS reads having approximately 14.6 times coverage [4]. It is also known to contain many repetitive subsequences.

The two most recent versions of the fruit fly genome have identical nucleotide sequences.<sup>3</sup> Thus, it was a good candidate for our research since both the reads and a complete and fairly stable finished sequence were available. A cutoff number of 40 (about  $2.5n$ ) was used for constructing the graph.

The Chicken, *Gallus gallus*, is the newest genome to be released, having been released in March of 2004 by the Washington University School of Medicine in St. Louis. Its 1 billion base pair genome was sequenced using approximately 6 times coverage by WGS reads. A cutoff number of 40 (about  $7n$ ) was used in constructing the graph. Since the chicken is a higher-order organism with a much larger genome, one might expect to find many more long repetitive elements. We did not, however, find many long repetitive elements. (See Chapter 4.)

---

<sup>3</sup>The difference between these two versions includes what subsequences of the genome the researchers consider to be genes.

## Chapter 4

### Results

#### 4.1 Characteristics of the Path-Builder

For each of the three sets of reads described in Chapter 3, we first created libraries of  $(k + 1)$ -mers with their corresponding frequencies (with  $k = 20$ ). Next, we used our path-building program to create paths through the resulting graph.

##### 4.1.1 Making Sound Choices

Recall that initially, there can be up to four outgoing edges for each node ( $k$ -mer) in the graph. Our goal is to create sequences that are representative of the most highly repetitive elements in the genome. For this reason, we eliminated all outgoing edges with the exception of the one with the greatest label (highest frequency). However, there are some nodes in the graph for which the label on two or more of its outgoing edges is the same or nearly the same.<sup>1</sup> If, for example, a node has two outgoing edges and one is labeled with a 10 and

---

<sup>1</sup>In the case of a tie, we retain the edge representing the  $(k + 1)$ -mer that occurs first alphabetically.



the other is labeled with a 12, then the corresponding  $(k + 1)$ -mers occur 10 and 12 times, respectively. Since the library of  $(k + 1)$ -mers is created from the reads, it is possible that this difference in frequencies is due to differences in the coverage of the genome by the collection of reads. We want to choose one outgoing edge over another based on the frequency of occurrence of the  $(k + 1)$ -mer in the underlying genome. Because the graph determines the paths and we want to faithfully represent the repeat regions in the genome, we need to feel confident that the percentage of nodes for which one or more edges have nearly the same label is nearly zero.

To answer this question, for each node we calculate the probability that, given a completely random selection from the four possible outgoing edges, the correct edge is chosen (where the correct edge is the one with the greatest label). For example, consider the node *ACGGT*. Suppose the total of the labels on all of its outgoing edges (before elimination) is 10, the edge representing *ACGGTA* is labeled 2, *ACGGTC* is labeled 3, *ACGGTG* is labeled 1, and *ACGGTT* is labeled 4. The edge representing *ACGGTT* has the greatest label and will remain in the final graph. The probability that this edge would be randomly chosen is  $\frac{4}{10}$ , or 40%. We say that this node has a **deterministic value** of 40%. We then look at all of the nodes in our graph and compute their deterministic values. Finally, we calculate the percentage of nodes in the graph that have a deterministic value greater than  $P\%$ , where  $P=50, 60, 70, 80, 90,$  and 100. We would like a very high percentage of the nodes to have a deterministic value above 90% because then we can be confident that our path-builder is faithfully representing the highly repetitive regions.

We look at the deterministic values for the graphs created from the reads

Deterministic Value	Percent of nodes (20-mers) by Genome		
	Chicken	Fruit Fly	Rat (BAC GSKF)
100%	$0 \times 10^{-4}$	$0 \times 10^{-4}$	$0 \times 10^{-4}$
90%	88	94.7	88
80%	91.8	97.2	94.6
70%	94.1	98.4	96.7
60%	96.3	99.2	98.8
50%	97.9	99.8	98.8

Table 4.1: This table gives the percent of nodes (representing 20-mers) in each genome’s graph having deterministic values of at least 100%, 90%, 80%, 70%, 60%, and 50%. There were 720064, 651950, and 19719 20-mers in the library of the chicken, fly, and rat, respectively. The rat BAC gsfk was selected for this table because it had the worst deterministic values of all of the 14 BACs.

for each genome and find that in each genome, approximately 90% of the nodes have a deterministic value of 90%. The graph in Table 4.1 shows for each genome studied the percentage of nodes in the graph that have deterministic values of 50%, 60%, 70%, 80%, 90%, and 100%. Based on the data given in Table 4.1, we conclude that in each of the genomes, the path-builder does not often make choices that depend on the random collection of reads covering the genome.

#### 4.1.2 Distribution of Path Types Across the Genomes

We can classify each  $k$ -mer in the library as belonging to one of four types of paths as in Section 2.2.1. One interesting question is, what percentage of  $k$ -mers in each library belong to a particular type of path? Table 4.2 shows the percentage of 20-mers in each genome’s library that belong to each type of path.

In all of the genomes, most of the  $k$ -mers belong to a terminating path. Since we are only looking at the most repetitive  $k$ -mers, we expect that these terminating paths represent repeat regions and the path terminates when it

leaves the repeat region and enters a unique region since the frequencies of  $k$ -mers in a unique region would fall below the cutoff. Periodic paths are sometimes representative of tandem repeat sequences that occur in the genome and such paths tend to have short period. However, we find that longer periodic paths are similar to long terminating paths in terms of how they match finished genomic sequence (see the next section). In this sense, we find that for long paths, the distinction between periodic and terminating may not be biologically important.

## 4.2 Matching Paths to Finished Sequence

Our reason for creating the path-builder is to make artificial paths that resemble repeat regions in the genome that are longer than those that would naturally be contained within a single read. Thus, a fair assessment of the success of our path-builder requires that we compare our artificial paths with repeat regions of the finished sequence in order to determine whether or not we are creating accurate representations of the repetitive elements in the genome. To do this, we use `blastn` [10], a program available through the National Center for Biotechnology Information (NCBI) that is designed to match strings of DNA

Path Type	Percent of 20-mers		
	Chicken	Fruit Fly	Rat (GBYZ)
Periodic	0.07%	1.4%	0.67%
Eventually Periodic	8.7%	13.5%	1.3%
Terminating	91.3%	85.1%	98%
Total No. of 20-mers	720,064	651,950	19,719

Table 4.2: The table above gives the percent of 20-mers in the graph of each genome that belong to a periodic, eventually periodic, or terminating path.

sequence against another DNA sequence. When we compare our long paths with the finished sequence, we often find several long, high fidelity matches. These matches are located at different places along the finished sequence and identify the locations of different copies of a particular repeat region in the genome.

### 4.2.1 The Rat

We used the rat genome to test our algorithm. For the BAC named gbyz, with 20-mers and a cutoff of 10 (about  $1.5n$ )<sup>2</sup>, our path-builder was able to build 982 paths up to 3920 letters long. (See Table 4.3.) The majority of our paths, however, tend to be fewer than one hundred letters long. Since we were only building paths with reads from a single BAC, the number of nodes in our graph is limited, thus our paths tend to be shorter. We found that many of the longer paths matched the finished sequence in several different places. For example, one path containing 2066 letters had over 60 high-fidelity matching subsequences with the rat genome (using blastn) [10]. These matching subsequences are all BACs and clones of the rat. The fact that a long path had so many high-fidelity matches with the finished sequence suggests that our path-builder is creating good representatives of the highly repetitive elements in the genome.

### 4.2.2 The Fruit Fly

Running the path-builder on the fruit fly with a cutoff of 40 (about  $2.5n$ ) created 50,244 paths. The longest path is 13,724 letters long. Forty paths are longer than 10,000 letters. (See Table 4.3.)

---

<sup>2</sup> $n$  is the average coverage of the genome by reads.

Since the fruit fly genome contains about 1000 times more letters than a rat BAC, the library for the fly was much larger; hence, longer paths could be built from the fruit fly reads than from the rat reads. Table 4.3 shows that 906 of the fruit fly paths are longer than 500 letters. A random sampling of 100 of these 906 paths were compared against the finished sequence of the fruit fly. We found that 74% have at least one high-quality match in the finished sequence (where a high-quality match is a subsequence that matches the path for at least 90% of its length with less than 10% error). An example of a path with multiple high-quality matches is discussed in section 4.3.

Furthermore, when these long paths were compared with multiple genomes using `blastn` [10], we found numerous high-fidelity matches with known **retroviruses** and **retrotransposons**. Retroviruses are viruses that infect their hosts by inserting a copy of their genome into the host genome. If inserted into the DNA of the sex cells, the retroviral DNA will be passed onto the offspring. Retrotransposons have been nicknamed “jumping genes” because they are sequences of DNA that are able to insert a copy of their sequence into a different section of the DNA. Thus, our path-builder was able to build these highly repetitive subsequences that appear in the genomes of many organisms using only reads from the genome of the fruit fly.

### 4.2.3 The Chicken

In building paths from the chicken reads, we used 20-mers with a cutoff of 40 (about  $7n$ ). There were approximately 93 million 20-mers in the library and file size restrictions forced us to use a high cutoff. The cutoff library contained approximately 720,000 20-mers. The path-builder did not build many long

	Rat	Fly	Chicken
Total No. of Paths	982	50244	142074
Path Type	Number of Paths		
Periodic	24	203	483
Eventually Periodic	27	2786	16514
Terminating	931	47255	125077
Length of Path	Number of Paths		
> 100 letters	146	2977	636
> 200 letters	90	2539	336
> 500 letters	64	906	5
> 1000 letters	24	633	0
> 10000 letters	0	40	0
Longest Path Length	3920	13724	577

Table 4.3: The table above highlights some interesting features of the set of paths built using 20-mers from the reads for each of the genomes.

Percent of nodes (20-mers) with no outgoing edges		
Chicken	Fruit Fly	Rat
23.9%	13.4%	5.2%

Table 4.4: The table gives the percent of nodes in each graph that have no outgoing edges. Notice that the chicken has the greatest percent of nodes with no outgoing edges. This is mainly due to the proliferation of  $N$ 's in the reads for the chicken. (An  $N$  is a low-confidence letter.)

paths from the chicken reads. Although there are 142,074 paths, only 636 are longer than 100 letters, and the longest path is only 577 letters—much smaller than the longest rat path. (See Table 4.3.) Table 4.4 gives the percent of 20-mers in the graph that have no outgoing edges.

One explanation for why few long paths were created from the chicken reads is that we used a high cutoff. We used a cutoff of  $2.5n$  for the fruit fly and a cutoff of  $7n$  for the chicken. Thus, fewer  $k$ -mers remained in the cutoff library and shorter paths were created. A second possibility is that the chicken genome does not contain many long, highly repetitive sequences. Although the chicken

is a much more complicated organism than the fruit fly, its genome is only about 5-10 times longer than the fruit fly's genome. Thus, it is plausible that the chicken genome has few, small non-coding regions of the genome.

Non-coding regions are where long repetitive elements are most commonly found; thus, it does not seem unreasonable to suggest (as our results do) that the chicken genome lacks long, highly repetitive sequences. However, as the sequence of the chicken genome remains largely unfinished, we must not make hasty conclusions about repeat regions in its genome.

### **4.3 An Example of a Successful Path in the Fruit Fly**

As previously mentioned, the reads for the fruit fly generated many more paths longer than 500 letters. Furthermore, many of these long paths match the finished sequence in multiple places. (See discussion in Section 4.2.2.) For example, a representative path 6839 letters long matches the finished sequence in 20 disjoint locations with less than 1% error. (An error is any discrepancy between the letters in the path and the letters in the matching subsequence. See table 4.5 for an explanation of errors.) Table 4.5 describes the matches for this path. The path has one matching subsequence in chromosome 2L that is 4138 letters long with no errors.

Notice that there are seventeen subsequences of the finished sequence that match the path for the entire length of the path. Of these seventeen subsequences, eleven have a 6-letter insertion, no deletions, and no substitutions. Upon closer inspection, we discover that the 6-letter insertion is

the same in each of the eleven matching subsequences; it is precisely the subsequence *AACTCT*. This subsequence of 6 letters usually occurs in the genome as a tandem repeat 24 letters long (4 side-by-side repeats). The tandem repeat never occurs as a set of 3 side-by-side repeats (18 letters) in the set of 20 matching subsequences. Since the path is built using 20-mers and the repeat is 24 letters, our path-builder produces only 3 tandem repeats of *AACTCT* (for a total of 18 letters) and then moves on to the letters in the genome that follow the 24-letter subsequence. Perhaps if we used 25-mers, we would be able to properly construct this string of 24 letters.

## 4.4 Distinguishing Between Copies of a Repeat Region

Since our path-builder often creates consensus paths through repeat regions, it may be possible to use our paths as a template against which we can compare multiple copies of a repeat region in the genome. If we find that one of our paths has several alignments (matches) with the finished sequence, none of which contain gaps, we can take advantage of the triangle inequality to compare these different matching subsequences with one another.

### 4.4.1 The Theory

Consider a path  $P$  having four distinct, gap-free alignments with the finished sequence. Label the aligning subsequences  $M_1, M_2, M_3$  and  $M_4$ . (See Figure 4.1.) Suppose further that  $M_1$  and  $M_2$  are identical sequences each having two bases that disagree with the path and  $M_3$  and  $M_4$  have three and



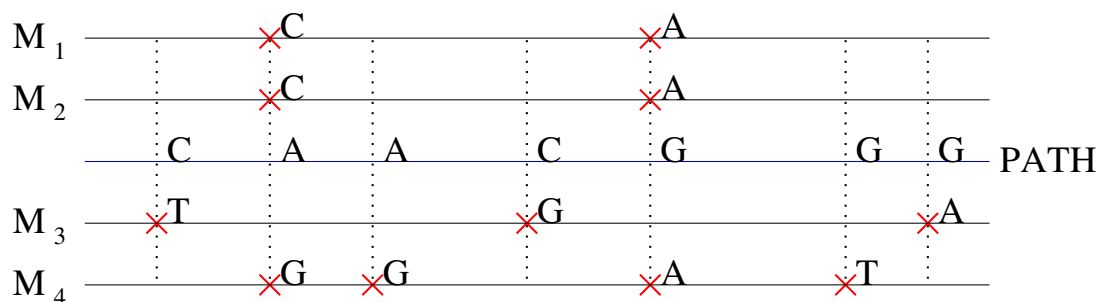


Figure 4.1: The path  $P$  is shown in the middle row. Mismatches between any of the four matching regions,  $M_1, \dots, M_4$ , and the path are shown as  $\times$ 's and labeled with the base. Regions of agreement are not marked. Aligning bases are highlighted with vertical dotted lines.

four disagreements, respectively, with the path. In order to distinguish between these four regions, we pairwise match each region against the other three regions. For this example, we will distinguish  $M_1$  from the other three regions.

Since  $M_1$  and  $M_2$  are identical sequences, there are no mismatching bases. Suppose that  $M_1$  and  $M_3$  have five mismatches and  $M_1$  and  $M_4$  have three mismatches as in figure 4.1. We then calculate the following ratio for each pair of matching regions, where  $E(A, B) =$  number of mismatches between  $A$  and  $B$ .

$$r_{ij} = \frac{E(M_i, M_j)}{E(M_i, P) + E(M_j, P)} \quad (4.1)$$

A ratio close to 1 suggests that  $M_i$  and  $M_j$  have independent mismatches with  $P$  and are different from one another. A ratio close to 0 suggests that  $M_i$  and  $M_j$  share similar mismatches with  $P$  and are nearly identical. A ratio of exactly 0 means that  $M_i$  and  $M_j$  are completely identical.

Returning to our example, we see that  $r_{12} = 0$ ,  $r_{13} = 1$ , and  $r_{14} = .5$ . These ratios suggest that  $M_1$  and  $M_2$  are completely identical;  $M_1$  and  $M_3$  have no conserved mismatches with the path and are distinguishable; and  $M_1$  and  $M_4$  share some of the same mismatches, but are still distinguishable from one

another. Thus, it seems that the finished sequence has a repetitive subsequence that occurs four times; two of the copies are identical ( $M_1$  and  $M_2$ ) and the other two copies are distinguishable from the two identical copies.

#### 4.4.2 Applying the Theory to Our Data Sets

We must determine whether or not this idea works well with real data. To test this theory, we choose a path of length 146 in the rat<sup>3</sup> that has 13 gap-free matching subsequences in the finished sequence; label them  $M_1, M_2, \dots, M_{13}$ . Then, we pairwise match each of these 13 subsequences and calculate  $r_{ij}$  for each pair of matches  $M_i, M_j$ . Using the information for each alignment, we can calculate  $r_{ij}$  for each pair (see Table 4.6). Based on these ratios, we can group the matches  $M_i$  according to how similar they are, using the value of  $r_{ij}$  to determine whether or not  $M_i$  and  $M_j$  are similar enough to be placed in the same group. Since we know that values of  $r_{ij}$  close to 0 indicate that  $M_i$  and  $M_j$  have similar disagreements with the path, we will group those pairs of matches for which  $r_{ij} < .40$ .

The value of .40 was chosen somewhat arbitrarily. Choosing a value closer to 0 would require that the grouped subsequences be more similar. However, one needs to be careful when choosing values closer to zero. It is possible that a pair  $M_i$  and  $M_j$  from the finished sequence have a few mismatching bases that are due to errors in sequencing, but are not actually errors in the two subsequences of the genome. Thus, we do not want to exclude possible matching subsequences solely because of sequencing errors. It should be noted, however, that if  $E(i, j) = E(i, P) = E(j, P)$  ( $M_i$  is as similar to  $M_j$  as

---

<sup>3</sup>This particular path was built from reads from the BAC gbyz.

it is to  $P$ ) then  $r_{ij} = .5$ . Based on a cutoff of  $r_{ij} = .4$ , we would group the matches as in Figure 4.2.

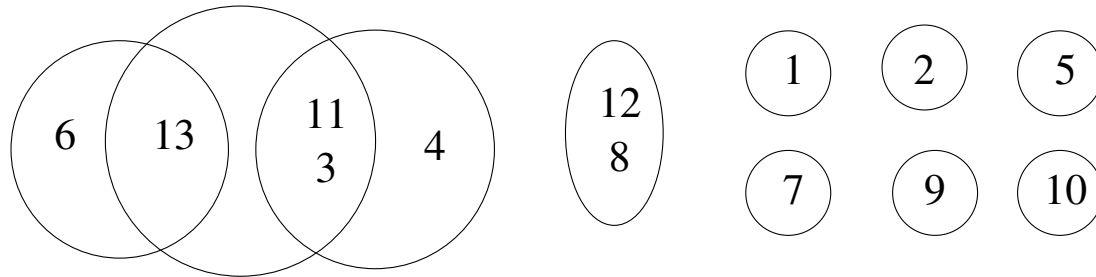


Figure 4.2: The figure above illustrates the groups that are created when we group those matches  $M_i$  and  $M_j$  for which  $r_{ij} < .4$ . Notice the interesting interaction between  $M_6, M_{13}, M_{11}, M_3$ , and  $M_4$ .

The results of this test suggest that we can use our paths to help identify and distinguish multiple copies of a repeat region in the genome using a relatively simple method. Section 5.1 suggests an application of this idea to assembly. More study is needed to determine which values of  $r_{ij}$  are best suited for use in assembly and in distinguishing multiple copies of a repeat region. Although it is not the goal of our present work, the values of  $r_{ij}$  and the groupings shown in Figure 4.2 may indicate biological consequences such as which copies of the repeat region were duplicated most recently.

Chromosome	Length	I	D	S	Percent Error
2L	6839	29	0	14	0.626%
2L	2003	6	0	2	0.398%
2L	6839	6	0	0	0.088%
2L	4138	0	0	0	0.000%
2R	6839	6	0	2	0.117%
2R	6839	6	0	0	0.088%
2R	6839	6	0	0	0.088%
3L	6839	6	0	0	0.088%
3L	6839	6	0	0	0.088%
3L	6839	6	0	0	0.088%
3L	6839	6	0	0	0.088%
3R	6839	8	0	18	0.380%
3R	6839	6	0	0	0.088%
3R	6839	7	0	0	0.102%
3R	6839	6	0	0	0.088%
3R	6839	6	0	0	0.088%
4	6839	6	0	1	0.102%
X	1555	2	2	5	0.579%
X	6839	6	0	1	0.102%
X	6839	6	0	0	0.088%

Table 4.5: The above table highlights 20 disjoint subsequences of the finished sequence that match an artificially created path of length 6839 letters for the genome of the fruit fly. These subsequences were selected because they each match the path for more than 1000 letters with a percent error less than 1%. The first column is the name of the chromosome containing the matching subsequence. The second column gives the length of the subsequence that matches the path. The third, fourth, and fifth columns outline the mismatches between the subsequence and the path. Insertions, I, are letters occurring in the finished sequence, but not in the path; deletions, D, are letters occurring in the path, but not found in the finished sequence; substitutions, S, are letters that disagree between the path and the subsequence. The percent error is the sum of columns 2, 3 and 4 divided by the length of the match. Notice that there is one perfect match of length 4138 in chromosome 2L.

	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>
<b>2</b>	.57											
<b>3</b>	.93	.95										
<b>4</b>	.89	.83	.3									
<b>5</b>	.85	.79	.8	.78								
<b>6</b>	.92	.95	.6	.67	.75							
<b>7</b>	.95	.89	.54	.59	.75	.67						
<b>8</b>	.87	.95	.43	.55	.83	.67	.71					
<b>9</b>	.65	.70	.89	.85	.75	.88	.94	.9				
<b>10</b>	.86	.90	.67	.70	.8	.6	.85	.71	.78			
<b>11</b>	.94	.82	.25	.33	.86	.71	.6	.56	.91	.75		
<b>12</b>	.87	.95	.43	.64	.83	.67	.57	.25	.9	.71	.56	
<b>13</b>	.93	.95	.14	.45	.83	.33	.43	.5	.9	.71	.33	.5

Table 4.6: The table above contains the values for  $r_{ij}$  for the rat path and its thirteen matches.

## Chapter 5

### Conclusions and Future Work

Determining the sequence of a genome is a difficult task for many reasons. One source of complication is the fact that the genome of any organism contains multiple nearly identical copies of a single repeat region. These copies of a repeat region tend to complicate assembly because reads belonging to different copies of the same repeat subsequence will overlap one another and, as a result, the different copies may be collapsed into a single contig during assembly. Thus, two disjoint regions of the genome will be assembled into one region. Correcting misassemblies after the assembly process is complete is difficult, time-consuming, and can sometimes require additional (expensive) laboratory techniques. Recognizing reads belonging to a repeat region and correctly assembling them is key to producing an accurate picture of the genome.

In an effort to understand what these repeat regions look like and to be able to identify them, we have created a deterministic algorithm that produces artificial subsequences of the genome (paths) from the reads. We create these paths using only the most frequently occurring  $k$ -mers in order to represent the most repetitive elements of the genome. The paths we create vary in several dimensions. First, the paths can terminate for three different reasons. Either a

path ends because it has encountered a  $k$ -mer that does not occur frequently enough, it falls into a periodic cycle, or it is itself a periodic cycle. Second, our paths vary in length. Lastly, our paths vary in how accurately they represent the genome. Some of our paths match the finished sequence in multiple places for the entire length of the path and others have very few or very short matches with the finished sequence. Initial tests with the rat and the fruit fly genomes suggest that our path-builder is creating good representatives of the long highly repetitive regions in the genome.

When a path greater than 500 letters long matches the finished sequence in multiple places for most of its length and there are no gaps in the alignment, we can calculate a ratio that gives an indication about the relationship between the different matching subsequences of the genome. These matching subsequences are potentially copies of the same repeat region. This ratio is close to 0 for a pair of matching subsequences when they share common mismatches with the path and are similar. When this ratio is close to 1, the pair of matching subsequences have independent mismatches with the finished sequence and are distinguishable. The results from running this test on a path in the rat suggest that this ratio can be used to group multiple matching subsequences by relative similarity and aid in distinguishing between different copies of the same repeat region. We may be able to use this method to help resolve repeat-induced problems in the assembly process.

## 5.1 Potential Applications to Assembly

During assembly, reads belonging to different copies of the same repeat region will frequently overlap one another. We may be able to use the paths we have

built as a template to group the reads according to the copy of the repeat with which they belong. The following is a method describing how we might do this.

### 5.1.1 Validation of Paths

Data from the fruit fly (refer to Section 4.2.2) suggests that approximately 74% of the long paths created using the path-builder have high-quality matches in the finished sequence. Our initial findings suggest that some of the longer paths may represent multiple repetitive elements rather than a single contiguous repeat region. In terms of applications, it would be best to be able to eliminate those paths that do not faithfully represent subsequences in the genome. One method of doing this is to compare the paths with the set of reads and eliminate any paths that do not match any of the reads. This validation step would create a set of **validated** paths (some of which would be smaller segments of longer paths) that are more likely to represent subsequences contained in the genome.

### 5.1.2 Avoiding Misassemblies due to Repeat Regions

When several reads overlap one another, call them  $M_1, M_2, \dots, M_n$ , we would find a validated path  $P$  that matches these  $n$  reads without gaps. (See Figure 5.1.) We would then match these  $n$  reads against  $P$  and pairwise against each other. We would calculate each  $r_{ij}$  and use this number to group the reads which are most likely to belong to the same copy of the repeat. We would limit the use of overlaps between pairs of reads to those belonging in the same group. Overlaps between reads in different groups would not be included in the assembly.



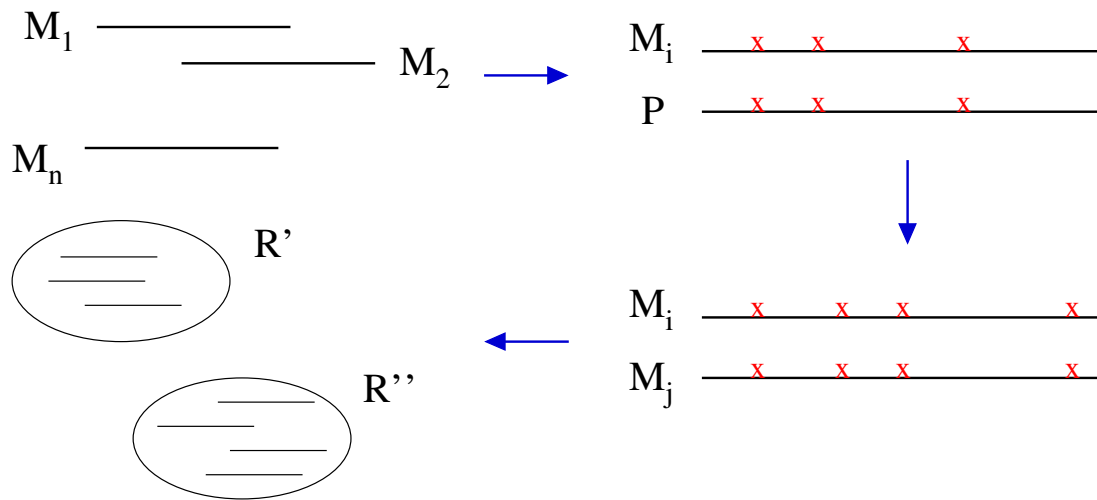


Figure 5.1: Overlapping reads  $M_1, M_2, \dots, M_n$  are first compared against the path  $P$ , then they are pairwise compared. Finally,  $r_{ij}$  is calculated for each pair and this value is used to group the reads according to the copy of the repeat  $R$  with which they belong.

## 5.2 So Where from Here?

Of course, the next question is “where do we go from here?” One obvious future goal is testing the proposed application to assembly outlined in Section 5.1. We would like to test this theory on problematic overlaps. This requires that we run tests to determine which values of  $r_{ij}$  are best suited for this application.

Currently, our reads for the rat are grouped by BAC. We would like to run our path-builder on the entire collection of reads for the rat (instead of running it separately by BAC) and see how successful it is.

Finally, we would like to determine what fraction of the reads is sufficient to create a library that will, in turn, produce paths that are representative of the majority of the repeat regions in the genome. Since time required to create the libraries increases with the increase in the number of reads, it would be very beneficial to be able to create the same paths using fewer reads.

### 5.3 Final Comments

According to results from our initial tests, the proposed algorithm for creating representatives of the highly repetitive subsequences of the genome has had decent success. We believe that the inclusion of a validation step would improve the resulting paths in terms of faithfully representing repeat regions. More work is needed, including testing the algorithm with other genomes; however, our initial tests indicate that this algorithm shows potential for being a powerful tool in genome assembly and in the identification of repeat regions in a genome.

## BIBLIOGRAPHY

- [1] P. HAVLAK, R. CHEN, K.J. DURBIN, A. EGAN, Y. REN, X. SONG, G.M. WEINSTOCK, AND R.A. GIBBS *The Atlas Genome Assembly System*, *Genome Research*, **14** (2004), 721-732.
- [2] *Introduction: Primer on Molecular Genetics*, Human Genome Project Information. <http://www.ornl.gov>.
- [3] G. MYERS *Whole-Genome DNA Sequencing*, *Computing in Science and Engineering*, **1** (1999), no. 3, 33-43.
- [4] E.W. MYERS, ET AL. *A Whole-Genome Assembly of Drosophila*, *Science*, **287** (2000), 2196-2204.
- [5] M. POP, S.L. SALZBERG, AND M. SHUMWAY *Genome Sequence Assembly: Algorithms and Issues*, *IEEE Computer*, **35** (2002), no. 7, 41-54.
- [6] RAT GENOME SEQUENCING PROJECT CONSORTIUM *Genome Sequence of the Brown Norway Rat Yields Insight into Mammalian Evolution*, *Nature*, **428** (2004), 493-520.
- [7] M. ROBERTS, W. HAYES, B. HUNT, S. MOUNT, AND J. YORKE *Reducing Storage Requirements for Biological Sequence Comparison*, Submitted.

- [8] M. ROBERTS, B. HUNT, AND J. YORKE *A Preprocessor for Shotgun Assembly of Large Genomes*, To appear in Journal of Computational Biology.
- [9] F. SANGER ET AL. *Nuceotide Sequence of Bacteriophage  $\lambda$  DNA*, Journal of Molecular Biology, **162** (1982), no. 4, 729-773.
- [10] D.J. STATES, W. GISH, S.F. ALTSCHUL *Improved Sensitivity of Nucleic Acid Database Searches Using Application-Specific Scoring Matrices*, METHODS: A companion to Methods in Enzymology, **3** (1991), no. 1, 66-70.