



CloneOrder : a clone ordering program for AFLP data
by Rebecca Lynn Lyle

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in
Computer Science

Montana State University

© Copyright by Rebecca Lynn Lyle (2002)

Abstract:

The biological sciences are becoming increasingly dependent on information science as they accrue large amounts of data that must be organized and analyzed in order to gain some new information or obtain results from the data. One such task is the physical mapping of genomes, which in this case, is creating a map that specifies the order of clones along a particular genome. Many genomes, especially grasses such as barley or wheat, contain retroposons and repetitive elements that make the task of building a physical map of the genome more difficult. The software project described in this thesis takes this into account and will allow these known problem segments of DNA to be filtered out before further calculations are done on the data. The algorithm's flexibility allows genomes with different degrees of variation to be pieced together with varying stringency levels chosen by the user, allowing more accurate maps to be created, based on the available data. The software will order the clones of large genomes quickly and accurately, displaying a graphical representation of the clones depicting the overlap and order as determined by the algorithm.

CLONEORDER: A CLONE ORDERING PROGRAM FOR AFLP DATA

by

Rebecca Lynn Lyle

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

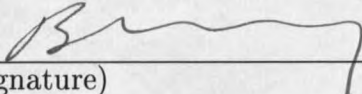
MONTANA STATE UNIVERSITY
Bozeman, Montana

November 2002

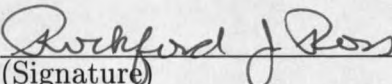
N378
298793

APPROVAL
of a thesis submitted by
Rebecca Lyle

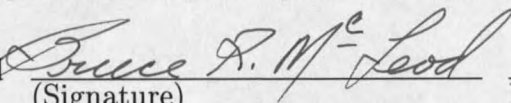
This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

Brendan Mumey  11/15/02
(Signature) Date

Approved for the Department of Computer Science

Rockford Ross  11/15/02
(Signature) Date

Approved for the College of Graduate Studies

Bruce McLeod  11-21-02
(Signature) Date

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under the rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Signature Rebecca Lyle

Date 11-08-02

This thesis is dedicated to my family. Thank you very much for your time and patience while I was working to complete my degree.

TABLE OF CONTENTS

ABSTRACT	ix
1. INTRODUCTION	1
2. BACKGROUND	3
Physical Mapping	3
AFLP/Fingerprinting	3
3. SOFTWARE ENGINEERING	9
System Requirements	9
Running Time	10
Memory	10
Algorithm	11
4. TECHNICAL ALGORITHM INFORMATION	12
Overlap Value Matrix	12
Scoring	15
Ordering	17
Overlap	18
5. SOFTWARE FREATURES	22
Using CloneOrder	22
Order Properties	24
Clone Length Display	24
Inline/Nestled Clone Display	25
Overlap Strictness	26
Clone Color	26
Reverse Order	26
Other Features	27
Slider Bar/Red Line	27
Drive-Over Clone Names	27
Save Order and Import Order	28
6. RESULTS	29
Ordering and Overlap	29
Running Time	33

7. CONCLUSION	35
Future Work	35
BIBLIOGRAPHY	37

LIST OF TABLES

Table	Page
1. Order and Overlap Results for 5x Coverage	30
2. Order and Overlap Results for 3x Coverage	31
3. Synthetic Data Time Trials	34
4. Real Data Time Trials	34

LIST OF FIGURES

Figure	Page
1. DNA Fingerprint created with the AFLP technique	4
2. Steps involved in AFLP Procedure	6
3. Fingerprint File	7
4. Fingerprint File Format	8
5. Symmetric Matrix	13
6. Lower Triangular Matrix	14
7. Overlap Value Matrix	14
8. Underlying Overlap Value Matrix	16
9. Ordered Overlap Value Matrix	16
10. Deviation Squared Matrix Example	19
11. Binary Overlap Matrix	20
12. Deviation Squared Matrix Example 2, Real Data Set 9	20
13. Binary Overlap Matrix Example 2, Real Data Set 9	21
14. Tolerance File Example	22
15. Order Properties Pop-Up Menu	24
16. Inline and Nestled Clone Display	25
17. Slider Bar, Size Line	27
18. Drive-Over Clone Name	28

ABSTRACT

The biological sciences are becoming increasingly dependent on information science as they accrue large amounts of data that must be organized and analyzed in order to gain some new information or obtain results from the data. One such task is the physical mapping of genomes, which in this case, is creating a map that specifies the order of clones along a particular genome. Many genomes, especially grasses such as barley or wheat, contain retroposons and repetitive elements that make the task of building a physical map of the genome more difficult. The software project described in this thesis takes this into account and will allow these known problem segments of DNA to be filtered out before further calculations are done on the data. The algorithm's flexibility allows genomes with different degrees of variation to be pieced together with varying stringency levels chosen by the user, allowing more accurate maps to be created, based on the available data. The software will order the clones of large genomes quickly and accurately, displaying a graphical representation of the clones depicting the overlap and order as determined by the algorithm.

CHAPTER 1

INTRODUCTION

In the plant genetics community, it is desirable to crossbreed different varieties of specific crops in order to produce new varieties with the best characteristics of multiple varieties. Desirable characteristics may include disease resistance, higher yield, protein content, and adaptability. Traditionally, crops are crossbred and grown to determine whether they exhibit the desired traits; only the desirable plants will then be reproduced. With recent scientific advances, plant genomes can be better characterized, and the genetic make-up of the plant can be determined. This information allows plant geneticists to screen new plants and discard those that do not contain genes for the desired traits. Early screening will greatly speed up the process of producing superior crops by crossbreeding, and will save money because undesirable strains will not have to be grown to maturity.

New technology being used in the genetics and molecular biology fields have made the type of data, and the amount of data that can be generated, too overwhelming to organize and interpret by hand. To generate a physical map using restriction enzyme fragments, hundreds of restriction fragment sizes from different clones have to be compared to each other to determine whether the clones overlap. The overlapping clones are then ordered so that each overlapping clone lies side by side, and non-overlapping clones are separated. The amount of overlap between clones is determined for each, and must conform with previously ordered clones in the contig, or ordered clone map. Manually ordering this type of data is tedious and time consuming especially considering laboratories today have several genomes they would like to further characterize in a short amount of time.

CloneOrder will use the AFLP[16] fingerprint data from several clones that are presumed to form a single contig and arrange these clones into the best ordering, displaying the clone overlap between pairs. This information will be used to determine which clones should be fully sequenced or further characterized. If a rough sequence of the entire genome is desired, then only the minimal tiling clones, determined by the CloneOrder display, will need to be sequenced. This will give minimal sequence coverage of the genome for which clones are available. Using the overlap results, additional clones can be chosen to resolve any ambiguous sections of the genome sequence. CloneOrder will also be useful if further characterization of specific sections of the genome is desired. The clone layout display will aid in choosing clones for positional cloning, isolating particular genes, as well as general data collection in order to learn more about specific genomes.

CHAPTER 2

BACKGROUND

Physical Mapping

Physical mapping of a genome is the determination of the physical location of landmarks or features along the genome. In this case, physical mapping refers to positioning clones in the correct relative order along the genome, so they may be selectively characterized. Physical mapping is important in many genome projects because the physical maps allow users to choose a specific segment of the genome to further study, or isolate segments associated with particular genes of interest.

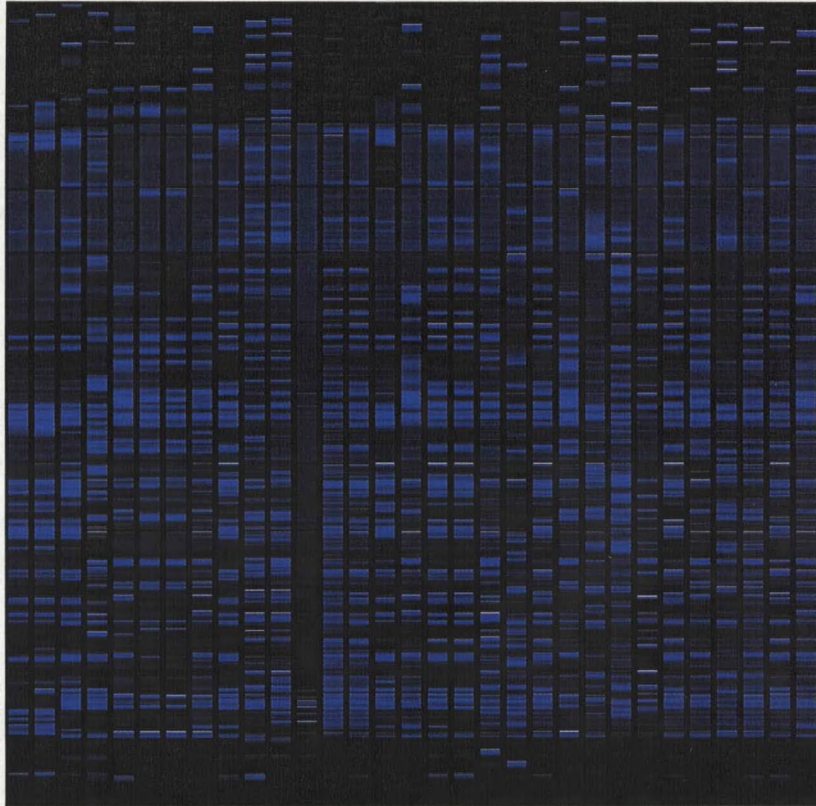
AFLP/Fingerprinting

A DNA fingerprint (Figure 1) is a display of DNA fragments, typically cut by restriction enzymes. DNA fingerprinting is used to show variation between individual genomes by depicting the differences in DNA sequence by their different fragment patterns. One technique for DNA fingerprinting is AFLP. This method is based on creating unique DNA fingerprints by cutting DNA with restriction enzymes, then amplifying¹ the fragments that will be used in the map creation step. These fragments are run out on a gel, and the resulting fingerprints are used to order the clones into contigs, or clone maps depicting the relative order of a group of overlapping clones that represent a genome, or genome section. Although the name AFLP should not be used as an acronym[16], it is often referred to as Amplified Fragment Length Polymorphism

¹DNA fragments are amplified, or copied many times, using Polymerase Chain Reaction (PCR) [17]

due to the similarity to a technique called RFLP, or Restriction Fragment Length Polymorphism.

Figure 1: DNA Fingerprint created with the AFLP technique



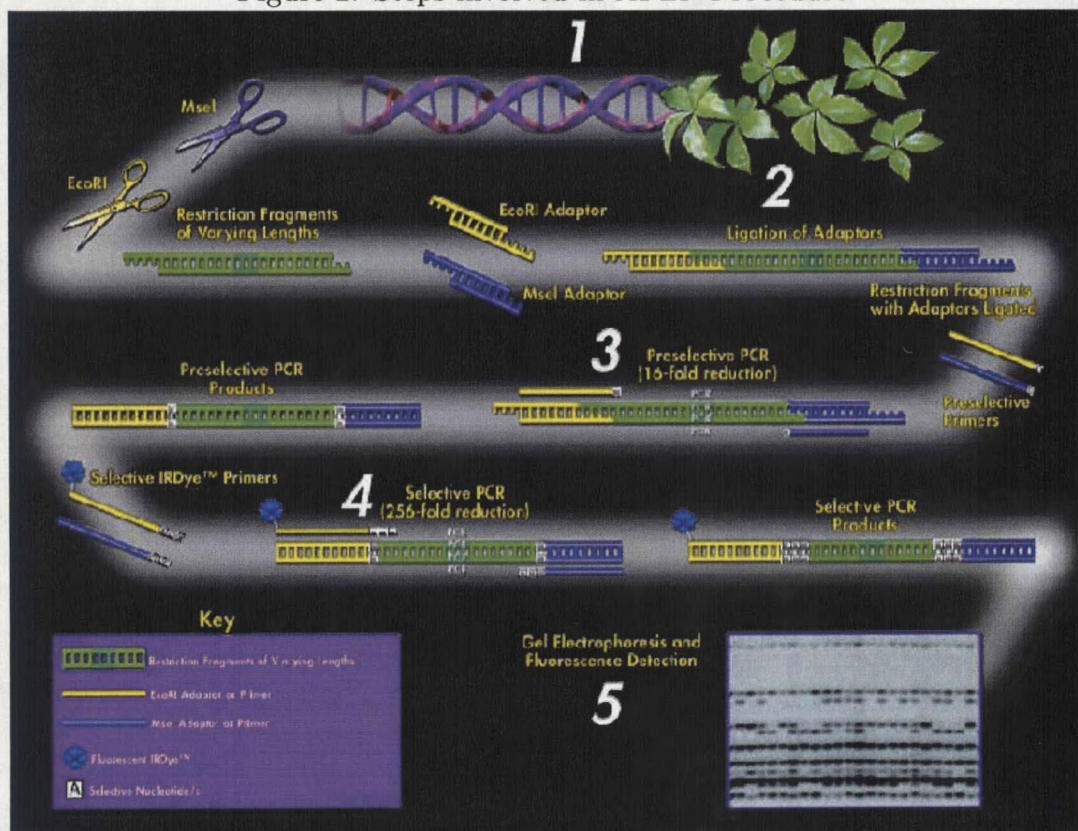
The AFLP technique was chosen here because it can be used for DNA of any origin and complexity, it requires no prior sequence information, and it can be performed with a limited amount of DNA. This is beneficial when trying to determine clone overlap and order in plants because many plant genomes are more complex than bacterial genomes, so the chosen method must provide accurate fingerprints despite this complexity. The DNA available for this project is derived from bacterial artificial

chromosomes (BACs). A BAC is a vector used to clone DNA fragments in *Escherichia coli* cells. The clone fragment size ranges from 100kb to 300kb, with the average insert size being about 150kb. The BAC clones obtained for this project average about 100kb, and there is no prior sequence information for these genomes. There is a limited amount of DNA available, making it difficult to use some common fingerprinting techniques such as Multiple Clone Digest Mapping (MCD Mapping)[3] where much more DNA is necessary.

Figure 2 shows a diagram of the steps involved in the AFLP method. The first step in the AFLP method involves digesting the DNA with restriction enzymes. Restriction enzymes cut the DNA at specific short sequences that are more or less randomly scattered throughout the genome. These short sequences are often palindromes, so both strands of the DNA are cleaved at nearly the same position. Typically, the DNA would be completely digested with a frequent cutter and a rare cutter. Adapters are then ligated to the ends of the fragments. Adapters are oligonucleotides, or short sequences of DNA, with a specific core sequence plus an enzyme specific sequence at the end that matches the sequence of the cut DNA. The enzyme specific sequence corresponds to the enzyme used in the restriction enzyme digestion. Attaching these adapters allows the DNA fragments to be replicated without ever having to know the underlying DNA sequence. The fragments are then replicated by polymerase chain reaction (PCR)[17] using AFLP primers that correspond to the adapters. The primers are short DNA segments that bind to the original DNA in a PCR reaction, creating a starting place for the DNA replication. Typically, two PCR reactions are performed: the first reaction is a pre-amplification of the fragments using less selective primers. This allows many fragments to be amplified. The second PCR reaction is a selective amplification, where the primers are extended using more selective nu-

cleotides. One of these primers is labeled (radioactive or dye-labeled) so the resulting fragments will be visible when they are run on acrylamide gels. The samples are then run out on a gel to separate the fragment sizes, producing a fingerprint of the genome. A typical fingerprint created with this technique will produce between 50 and 100 amplified fragments. Overlapping regions in the clones can be detected by the presence of similar sized fragments. Since the fragments originate from the same segment of the genome, they will contain the same restriction sites, and therefore the restriction fragments will be of similar size.

Figure 2: Steps involved in AFLP Procedure



Copyright LI-COR Biosciences, used with permission. AFLP is a registered trademark of Keygene, N.V.

After separating the fragments of each of the clone samples on a gel, the fragment size information must be extracted for each gel lane. Each lane in the gel corresponds to a single clone. Using a software program such as Genographer[?], the lanes of a gel may be imported, and a visual depiction of the fingerprint is shown. The fragment size information can be exported as a fingerprint file (Figure 3) containing the fragment sizes for each gel lane imported. The input for the CloneOrder software program is this set of fingerprints from clones that are believed to originate from the same genome. Figure 4 shows the accepted input format for a fingerprint file. In this format, the first line contains the number of clones, and the number of independent restriction enzyme digests. The following line contains the name of the clone, and the number of fragments from each restriction enzyme digest. The next lines list each of the fragment sizes measured for each restriction enzyme digest, with each new digest beginning on a new line. This information is repeated for each clone.

Figure 3: Fingerprint File

```
8 1
clone0 14
211 75 102 373 43 85 671 198 1033 387 64 119 36 603
clone1 13
334 43 85 671 198 1033 387 64 119 36 664 151 215
clone2 13
326 43 85 671 198 1033 387 64 119 36 664 151 223
clone3 12
70 198 1033 387 64 119 36 664 151 849 76 353
clone4 11
697 387 64 119 36 664 151 849 76 358 599
clone5 11
351 64 119 36 664 151 849 76 358 707 625
clone6 11
304 64 119 36 664 151 849 76 358 707 672
clone7 9
460 151 849 76 358 707 1018 344 37
```

Figure 4: Fingerprint File Format

```

<number of clones (m)> <number of independent restriction digests (n)>
<clone name 1> <# fragments in digest 1> <# fragments in digest 2> ... <# fragments in digest n>
<digest 1: fragment length 1> <fragment length 2> ... <fragment length n>
<digest 2: fragment length 1> <fragment length 2> ... <fragment length n>
.
.
<digest n: fragment length 1> <fragment length 2> ... <fragment length n>
<clone name 2> <# fragments in digest 1> <# fragments in digest 2> ... <# fragments in digest n>
<digest 1: fragment length 1> <fragment length 2> ... <fragment length n>
<digest 2: fragment length 1> <fragment length 2> ... <fragment length n>
.
.
<digest n: fragment length 1> <fragment length 2> ... <fragment length n>
.
.
<clone name m> <# fragments in digest 1> <# fragments in digest 2> ... <# fragments in digest n>
<digest 1: fragment length 1> <fragment length 2> ... <fragment length n>
<digest 2: fragment length 1> <fragment length 2> ... <fragment length n>
.
.
<digest n: fragment length 1> <fragment length 2> ... <fragment length n>

```

CHAPTER 3

SOFTWARE ENGINEERING

CloneOrder was developed for a specific problem described by the Plant Genetics laboratory at Montana State University. The laboratory acquired some BACs, from which they wanted to create a physical map in order to limit the number of clones they would need to further characterize. They were mainly interested in determining which clones overlapped in sections of the genome, so a minimal number of clones could be used in further tests. Fingerprints of the BACs were being created using the AFLP technique, but they had no software available to order the clones based on this information. Ordering, or even determining overlap, by hand is tedious, and somewhat arbitrary work. Looking at the fingerprints, the overlap can be suggested by noticing many common fragments, but ordering these clones based on the visible fingerprints is very difficult. Many plant genomes also contain sections where retroposons or repetitive elements complicate the ordering and overlap because the same DNA sequence may appear in several sections of the genome. If a genome is known to contain these elements, and if they produce fragments of a particular size, the fragment size may be filtered from the fingerprint file using the CloneOrder software. Filtering problem fragment sizes prior to ordering the clones allows a more accurate map to be created.

System Requirements

The software has been tested on computers running both a Windows and Linux operating system; it should be compatible with any operating system that supports the Java Runtime Environment (JRE). Java was chosen as the language to write this

software for portability reasons. Since the algorithm itself is fairly fast, speed was not an issue in language consideration. Swing Utilities were used in the development, so JRE 2.0 or above must be installed in order to run CloneOrder. The speed of the algorithm allows it to be run with a relatively slow central processing unit (CPU), and still get results in a reasonable amount of time. Due to the amount of information stored during the calculations, the amount of memory necessary is dependent on the genome size, and the number of clones being ordered. Typically 64 MB of RAM will be sufficient, but more memory may be necessary for very large or complex genomes.

Running Time

Despite the number of calculations involved, the actual running time is fairly fast. A typical running time to calculate the overlap values on a 500Mhz computer is between one and two seconds, depending on the size and complexity. The time required to order the clones is less than two seconds. This is only the clock time, it doesn't take into account any system induced waits that may have occurred.

Memory

It is suggested that at least 64MB of memory is used to run CloneOrder. This should allow most large or complex genomes to be run without memory problems. More memory may be necessary to run multiple orderings, or extremely large or complex genomes with many clones. The minimum amount of memory needed is 16MB. This is sufficient to order a test data set with 25 clones, and a genome length of 400,000 base pairs.

Algorithm

The original algorithm and idea for this project was based on a Doctoral dissertation by Brendan Mumey, *A Powerful Clone Overlap Test for Restriction Fragment Mapping* [13]. After designing the software and testing it on several real data sets, it was apparent that the problem set was not similar enough to that used in Mumey's thesis, so the algorithm had to be re-written to fit the problem set available here.

CHAPTER 4

TECHNICAL ALGORITHM INFORMATION

The algorithms created for CloneOrder determine both the clone order and overlap based on the assumption that each of the common fragments in the clones are actually pieces from the same segment of the genome. This assumption can be made because the method used to create and measure the restriction fragments is very accurate; fragments with sizes under 500 base pairs can be measured accurately within fractions of a nucleotide[8]. If multiple clones share a number of fragments, or a significant percentage of their length, it can be assumed that these clones overlap, and that the similar fragments are actually copies of the same segment of the genome. The final clone ordering is derived from these assumed clone overlaps.

Overlap Value Matrix

An overlap value matrix is created to record the confidence that any pair of clones overlap with each other. Overlap between clones is determined by both the number of restriction fragments that overlap and the length of the overlap. Considering the number of overlapping restriction fragments is important because the greater the number of common fragments, the more likely it is that the clones actually overlap, and the less likely the similarities were by coincidence. The overlapping length between clones is valuable because several small restriction fragments could appear to be overlapping between clones, when in actuality they are just random small segments of DNA. If the fragments comprise a significant length of the clone, they are more likely valid fragments, and the clones are more likely to overlap.

The overlap value matrix consists of rows and columns, where each row and corresponding column represent a clone. The overlap value matrix is a symmetric matrix (Figure 5), meaning the values above the diagonal are equal to those below the diagonal. Therefore, it is only necessary to use half of the overlap value matrix. This algorithm uses the overlap value matrix values from the diagonal down to the lower left corner, making it a lower triangular matrix (Figure 6). The matrix cells contain an overlap value describing the chance that any two clones overlap with each other. This value is based on the number of restriction fragments the two clones share in common compared to the total number of restriction fragments in each clone, as well as the total length of the fragments in common compared to the total length of fragments of each individual clone. If multiple enzymes are used to independently digest the genome, each is considered and the average overlap value score for all enzymes is placed into the overlap value matrix. This overlap value is calculated for each pair of clones and the resulting values are placed in the overlap value matrix (Figure 7). Each clone has an overlap value of 100%, or 1.00, of overlapping with itself, therefore a "1.00" is placed in each cell on the diagonal. This overlap value matrix is used in the scoring and ordering of the clones.

Figure 5: Symmetric Matrix

1.00	0.62	0.53	0.05
0.62	1.00	0.82	0.14
0.53	0.82	1.00	0.15
0.05	0.14	0.15	1.00

Figure 6: Lower Triangular Matrix

1.00			
0.62	1.00		
0.53	0.82	1.00	
0.05	0.14	0.15	1.00

Figure 7: Overlap Value Matrix

	clone0	clone1	clone2	clone3	clone4	clone5	clone6	clone7
clone0	1.00							
clone1	0.66	1.00						
clone2	0.66	0.85	1.00					
clone3	0.46	0.65	0.65	1.00				
clone4	0.24	0.43	0.43	0.64	1.00			
clone5	0.15	0.34	0.34	0.55	0.65	1.00		
clone6	0.15	0.34	0.34	0.55	0.65	0.79	1.00	
clone7	0.00	0.07	0.07	0.28	0.38	0.52	0.52	1.00

The following formula is used to calculate the overlap value for each clone pair:

$$\text{Overlap Value} = \frac{\sum_{k=1}^{\text{total enzymes}} \left[\frac{\left(\frac{\# \text{ matching fragments}}{\# \text{ fragments A}} \right) + \left(\frac{\text{matching length A}}{\text{total length A}} \right)}{2} + \frac{\left(\frac{\# \text{ matching fragments}}{\# \text{ fragments B}} \right) + \left(\frac{\text{matching length B}}{\text{total length B}} \right)}{2} \right]}{\text{total enzymes}}$$

The overlap value is calculated by first taking the sum, for each enzyme, of the average of the ratio of matching fragments to total fragments for clone A and the ratio of length that matches for clone A to the total length of clone A plus the average of the ratio of matching fragments to total fragments for clone B and the ratio of the matching length for clone B to the total length of clone B. The average of this number is taken, and then divided by the number of enzymes used to get the average overlap value for all enzymes.

Scoring

The scoring function is used to determine the score of a particular clone permutation, or a possible clone ordering. A given permutation is represented by the ordering of the rows and columns in the overlap value matrix. The matrix is scored, and the score associated with the given ordering is returned to aid in the ordering of the clones.

The scoring function (Algorithm 1) is based on weighted values from the overlap value matrix. The weight applied to a given overlap value is inversely proportional to its distance from the diagonal in the overlap value matrix. This gives more credibility to the values closer to the diagonal. Values that fall further from the diagonal are discounted, but not disregarded, so these values have less influence on the score. If the clones are in the proper order, overlapping clones will be adjacent to the clone they overlap with, or to a group of overlapping clones. Clones that overlap should have higher overlap values; these values will fall near the diagonal when the clones are ordered correctly. This weighting of the values in the scoring is done so clones in a more likely order will have higher scores than unordered clones.

Algorithm 1 Scoring the Overlap Value Matrix

First, step through the matrix columns, and find the zero's.

Set any values that fall below a zero in a column to zero (discarded).

Next, calculate the score/cost using the following algorithm:

```

for k = 1 to size
  for i = k to size
    j = i - k
    score = score + (matrixValue[i * size + j] / k)
  return score

```

The score is calculated by taking the sum of the overlap value matrix values, weighting the values closer to the diagonal more than those nearer to the corner. When the clones are ordered, this will force all of the higher values to migrate to the diagonal, resulting in a higher score (Figure 8 and Figure 9). Once a zero is encountered in a column, the rest of the overlap value values in that column are discarded so they will not artificially improve the score (Figure 9). A zero in a column represents a discontinuity (zero overlap). If the clones are in the correct order, each cluster of overlapping clones will be separated by a discontinuity, or zero, in the matrix. This zero-ing of the columns also helps force the higher values to the center because these values are only considered in the final score if they fall before a zero.

Figure 8: Underlying Overlap Value Matrix

1.00					
0.00	1.00				
0.00	0.57	1.00			
0.00	0.18	0.18	1.00		
0.00	0.00	0.00	0.14	1.00	
0.00	0.14	0.14	0.00	0.53	1.00

Figure 9: Ordered Overlap Value Matrix

1.00					
0.18	1.00				
0.18	0.57	1.00			
0.00	0.14	0.14	1.00		
(0.14)	0.00	0.00	0.53	1.00	
0.00	0.00	0.00	0.00	0.00	1.00

The (0.14) will become zero since a 0.00 precedes it in column 1

Ordering

The ordering algorithm finds the permutation of the clones that has the highest score. The score of the permutation is determined by the scoring algorithm. Although the true ordering may not have the absolute highest score, this is the best approximation that could be made here.

Ordering is accomplished using a series of greedy searches with each clone, in turn, being used as the starting clone for the search. From the starting clone, the next clone is chosen by determining which of the remaining unordered clones has the highest overlap value with the initial clone. This clone is chosen as the next clone in the ordered sequence. The process is repeated with the next clone as the comparison point. This is repeated until all clones are ordered. The resulting permutation is scored. The order with the highest score from all derived orders is chosen as the best ordering.

To better visualize the ordering algorithm (Algorithm 2), each clone can be represented as a node in a graph. The graph is fully connected, so each node is connected to every other node by an edge. Each edge has a value, though the value may be zero. It is assumed that since there is some order to the clones, that there will be a path through the clones. When a node is chosen for the ordering, it is said to be visited. Each node may only be visited once, and no node may remain unvisited. Each node, in turn, is chosen as the starting node. From the starting node, the edge value for each unvisited node is checked, and the node connected to the edge with the highest value is chosen. This new node is the new pivot point from which the next edge is checked, and the attached node is the next in the ordering. This process is repeated until all nodes have been visited. The resulting order is then scored, and

the permutation with the highest score is chosen as the best ordering. If multiple orderings result in the same score, one permutation is arbitrarily chosen.

Algorithm 2 Graph Clone Ordering

Construct a fully connected graph such that:

each node represents a clone

each edge contains the overlap value associated with the nodes it connects

For each node i in the graph

mark all nodes as being unvisited

mark node i as the first node in the order

Repeat until all nodes have been visited

mark the current node as being visited

identify the longest edge connecting the current node to an unvisited node

mark the unvisited node connected to that edge as the next clone in the ordering

Overlap

The determination of overlap between clones is considered after the final ordering is chosen. Whether or not there is overlap is determined by the deviation squared value for each pair of clones. A deviation squared matrix (Figure 10) is created to store these values for each set of clones. The deviation squared value is the sum of the squared deviations from the sample mean of each value in the column up to, and including the current value. The deviation squared value is found using the formula $\sum_{i=1}^k (x_i - \bar{x}_k)^2$. This value is also known as the sum of squares value; it reflects the degree which numbers in a set vary among themselves. When there is no variability, the deviation squared value will be zero, but as the variation between numbers in the set increases, the squared deviation value will also increase. The deviation squared value is used because it is easily calculated for each new value associated with the

clone pairs, and it clearly differentiates values that substantially stray from the sample mean, while still accepting lower values which may only indicate less overlap, but not lack of overlap, between clones. Once a zero is encountered in a column, all other values are disregarded. A zero in a column indicates no overlap, or a separation of clusters. In this case, a '9.00' is placed in the matrix cell when the value should be disregarded.

Figure 10: Deviation Squared Matrix Example

1.00					
0.33	1.00				
0.45	0.09	1.00			
9.00	0.37	0.37	1.00		
9.00	9.00	9.00	0.11	1.00	
9.00	9.00	9.00	9.00	9.00	1.00

The cutoff for accepting or rejecting a deviation squared value as an overlapping value is determined by the strictness level chosen by the user. The suggested value, as well as those for each level of strictness were chosen empirically based on real data sets. The actual deviation squared value has less meaning than its representation as a place where a drastic change in overlap values occurs.

The results of the overlap determination for each pair of clones is stored in a binary overlap matrix (Figure 11). Figure 12 and Figure 13 show a deviation squared matrix and a binary overlap matrix, respectively, for a small real data set. These examples more clearly show how the deviation squared values are used to create the binary overlap matrix. In this example, the standard cutoff value of 0.40 was used; any clone pairs with deviation squared values greater than the cutoff are considered non-overlapping.

Figure 11: Binary Overlap Matrix

1					
1	1				
0	1	1			
0	1	1	1		
0	0	0	1	1	
0	0	0	0	0	1

Figure 12: Deviation Squared Matrix Example 2, Real Data Set 9

1.00													
0.10	1.00												
0.14	0.00	1.00											
0.19	0.13	0.10	1.00										
0.22	0.20	0.13	0.00	1.00									
0.24	0.25	0.15	0.03	0.02	1.00								
9.00	9.00	9.00	0.13	0.11	0.12	1.00							
9.00	9.00	9.00	0.52	0.50	0.39	0.11	1.00						
9.00	9.00	9.00	0.82	0.76	0.55	0.29	0.29	1.00					
9.00	9.00	9.00	0.98	0.86	0.60	0.30	0.44	0.30	1.00				
9.00	9.00	9.00	1.09	0.92	0.63	0.30	0.49	0.43	0.01	1.00			
9.00	9.00	9.00	1.23	0.97	0.67	0.30	0.57	0.50	0.03	0.01	1.00		
9.00	9.00	9.00	1.29	1.03	0.69	0.30	0.60	0.53	0.04	0.02	0.02	1.00	
9.00	9.00	9.00	1.38	1.11	0.73	9.00	0.63	0.55	0.06	0.05	0.06	0.09	1.00

Once the overlap matrix has been created, the amount of overlap must be determined for each overlapping pair. This amount is determined by taking the average of the sum of all matching fragments for each enzyme. When multiple clones overlap, the amount of overlap between each clone is considered, and the average overlap amount is used in the display. This gives an accurate depiction of the clone overlap when the display clone length is viewed as the "Sum of Fragments". When the "Average Clone Length" view is chosen, the previously calculated overlap amount is used

to calculate a percentage of the total clone length that overlaps, and that percentage of the supplied clone length is displayed as the overlapping section of the clone.

Figure 13: Binary Overlap Matrix Example 2, Real Data Set 9

1														
1	1													
1	1	1												
1	1	1	1											
1	1	1	1	1										
1	1	1	1	1	1									
0	0	0	1	1	1	1								
0	0	0	0	0	0	1	1	1						
0	0	0	0	0	0	0	1	1	1					
0	0	0	0	0	0	0	1	0	1	1				
0	0	0	0	0	0	0	1	0	0	1	1			
0	0	0	0	0	0	0	1	0	0	1	1	1		
0	0	0	0	0	0	0	1	0	0	1	1	1	1	
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

CHAPTER 5

SOFTWARE FEATURES

Using CloneOrder

Fingerprint data is imported into the CloneOrder program using the 'Import' command. This makes the data available to be used and/or modified. The user is also asked to supply a tolerance file. The tolerance file (Figure 14) contains information about the accuracy of the fragment measurements. Although it is possible to measure the fragment sizes with extreme accuracy, changes in the technique, equipment, or expertise of researchers may introduce some error into this measurement. The tolerance file is used to account for this error.

Figure 14: Tolerance File Example

11	
316	0.04
562	0.03
1000	0.025
1778	0.022
3162	0.021
5623	0.022
10000	0.024
13335	0.027
17780	0.04
23714	0.06
31620	0.10

The tolerance file contains error percentages for each range of restriction fragment sizes allowing the algorithm to consider two fragments equal if they are within the

allowed deviation. Figure 14 shows the accepted file format for a tolerance file. The first line contains the number of tolerances in the file, and the next lines contain the largest fragment size, and associated error percentage. Looking at figure 14, we see that any fragments up to and including 316 base pairs may contain a 4% error. Fragments from 317 base pairs to 562 base pairs may contain a 3% error. These error rates are only examples, and are specific to each laboratory.

The next step depends on whether there are known problem sizes in this genome. If the genome contains retroposons, repetitive elements, or other known problem fragment sizes, they should be filtered out before calculations are performed. Using the 'Filter' command, the user can either add filters manually, or add a filter file containing the fragment sizes to be removed. Once the desired filter is added, the user will choose the 'Filter Data' command to remove the problem fragment sizes from the imported fingerprint. This will not alter the original fingerprint file, but instead will create a filtered fingerprint file which will be used in subsequent calculations.

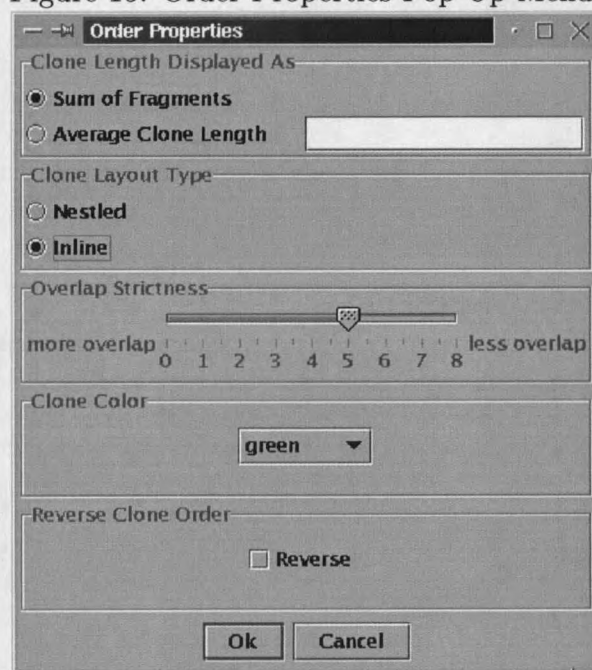
After the fingerprints have been imported and if necessary, filtered, the overlap value information must be calculated. This is done by choosing the 'Calculate Probabilities' command. Since this step is often the most time consuming computationally, it is done separately, and the information is saved in a file for future use by the ordering portion of the program. This step may not be necessary in subsequent orderings of the same genome, provided no changes have been made.

The ordering portion of the program is run by choosing the 'Order Clones' command. This step uses the overlap value information to place the clones in the most likely ordering. After the ordering calculations have been performed, the clones are graphically displayed, showing the overlap between pairs of clones.

Order Properties

The 'Order Properties' pop-up menu allows the user to change the display of the clone layout. This may be necessary due to differences in the complexity of the genome, the information desired, or personal preferences.

Figure 15: Order Properties Pop-Up Menu



Clone Length Display

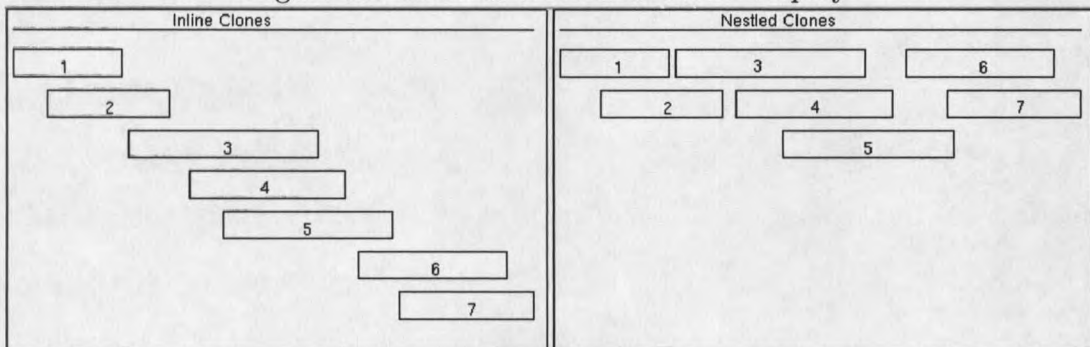
In the AFLP technique, the sum of the fragments in a clone is not the same as the total clone length. This is because many of the fragments do not fall within the chosen size range, or selection criteria, and are not included in the AFLP data. For this reason, the clone lengths may be displayed in two ways; the length may be shown

as the sum of the fragment lengths, or it may be displayed as an average clone length supplied by the user. The average clone length display shows a more realistic overall picture, but does not differentiate the number of fragments contained in each clone. The “Sum of Fragments” clone length display clearly differentiates clones that contain very few fragments from those with many fragments, making it easier to judge the validity of each clone in the contig.

Inline/Nestled Clone Display

The user also has the choice of displaying the clones ‘inline’ or ‘nestled’ (Figure 16). The inline display lines the clones up, one per line, with the starting position offset to show overlap between clones. This may be useful if you were interested in clones around a particular section of the genome, or if the actual order was more important than the overlaps. The nestled layout is more common, and it shows neighboring clones placed on the same line, next to each other, unless they overlap. Overlapping clones are drawn on the next available line, assuring the that full length of each clone is visible. This layout clearly displays the clone coverage of any section of the genome, as well as neighboring clones to a particular segment of the genome.

Figure 16: Inline and Nestled Clone Display



Overlap Strictness

The overlap strictness is one of the most important user choices in the Order Properties menu. The overlap strictness slider allows the user to adjust the cutoff that determines whether two clones overlap or are disjoint. This option is useful to account for genome complexity, and to adjust the accuracy desired in the final map. If higher strictness levels are chosen, less overlap will show between clones, but those that are overlapping will have a higher certainty of overlap. If lower strictness levels are chosen, the algorithm is much more lenient in its definition of overlap, and some clones could possibly be shown to overlap when in actuality, they should be disjoint. A lower strictness level is useful if a batch characterization is going to be done, and it would be easier to test any possibly overlapping clones, rather than go back and re-do them later. The suggested level of strictness is a "5"; this allows the most overlap, with the least errors for typical plant genomes.

Clone Color

The clone color option is given for personal preference. This changes the display color for the clones in the contig map.

Reverse Order

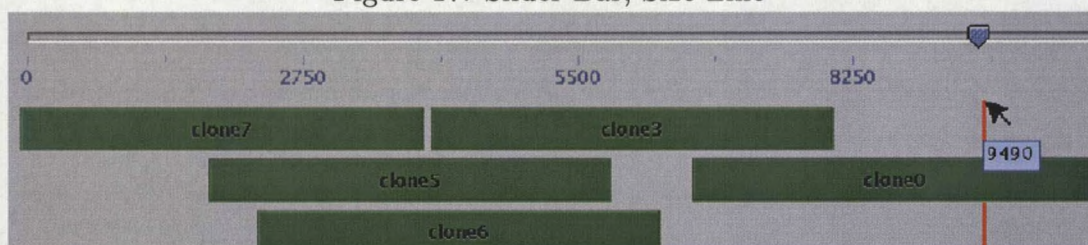
Since the algorithm doesn't have any information about the starting or ending clone; one clone is chosen and the others are ordered appropriately following that clone. This could result in a reversed map, where the last clone is displayed as the first clone. If a partial ordering, or the start of the contig is known, and the map appears to be reversed, it may be easier to visualize if all of the clones were placed in the correct direction. Choosing the "Reverse" option simply reverses the current ordering, allowing the user to visualize the clones in a more accurate physical map.

Other Features

Slider Bar/Red Line

The slider bar is located above the ordered clones in the final map (Figure 17). By dragging the pointer on the slider bar, a red line will appear and can be moved along the genome. Place the red line in a desired location and hold the mouse arrow over the line, and the base pair size at that location will be displayed. This red line can be used to visually identify coverage of any section of the genome, determine overlap on closely located clones, and to determine the size or location of specific clones along the genome.

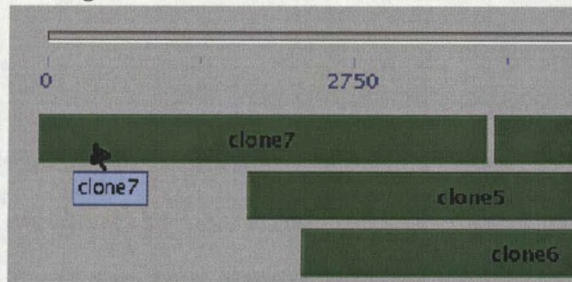
Figure 17: Slider Bar, Size Line



Drive-Over Clone Names

Occasionally a clone's display will be too small to fit the entire clone name inside. In this case, the name can be determined by placing the mouse arrow over the clone; the clone name will then be displayed (Figure 18).

Figure 18: Drive-Over Clone Name



Save Order and Import Order

If a clone map is created that the user would like to refer to at a later time, the ordering may be saved, and imported at a later time. To use this option, create the map you would like to save, then simply choose 'Save' or 'Save As', and save the order as an ".ord" file. To re-display the order, simply use the "Import Order" command and import the appropriate order file (.ord).

CHAPTER 6

RESULTS

Ordering and overlap tests were run on both synthetic data and real data sets. The real data provided a more accurate view of the algorithm's performance because the data was created using the methods this software was designed to utilize. The true order and overlap were not known for the real data sets, so performance in the real data sets was judged by visually inspecting the fingerprints for similarities between clones. Synthetic data sets with known orderings and overlap amounts were used to test the accuracy of the ordering and overlap algorithm.

Ordering and Overlap

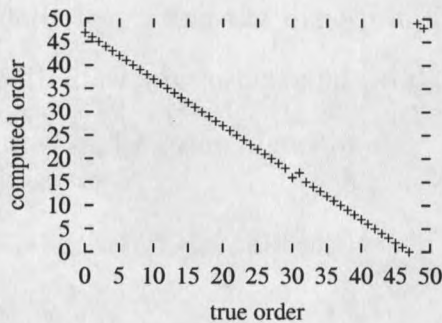
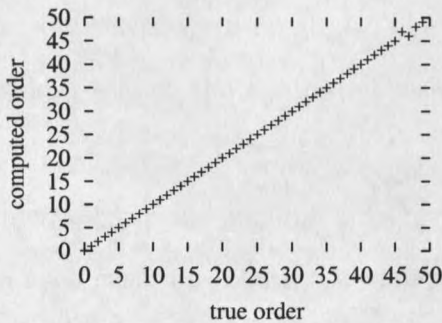
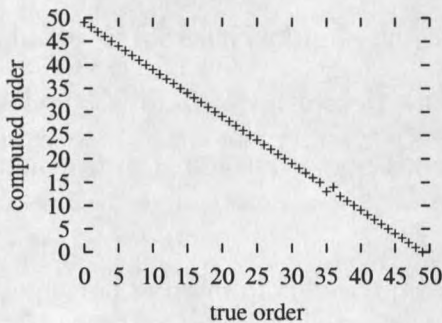
In order to test the accuracy of the ordering and overlap results produced by CloneOrder, several synthetic data sets were created. The true order and overlap amounts are known for each data set, so the computed results can be compared to the known data.

Several tests were performed on various genome sizes, with a varying number of clones and clone coverage. In each case, no tolerances were included in the tolerance file, so only exactly matched fragments were considered matching. The strictness level was set at the default value of '5' for each example shown. The average fragment length (shown as "Avg. Frag. Length" in Tables 1 and 2) was changed for each set to show that a particular fragment length is not necessary; the difference between them was negligible.

The clone coverage is the expected number of clones representing any given section of the genome. A clone coverage of 5x indicates 5 clones are expected to represent

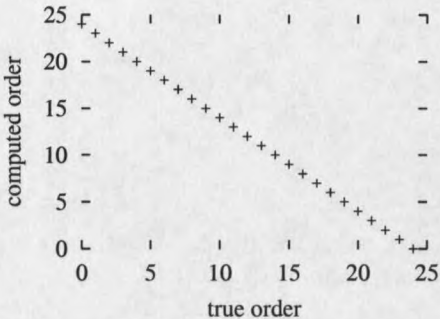
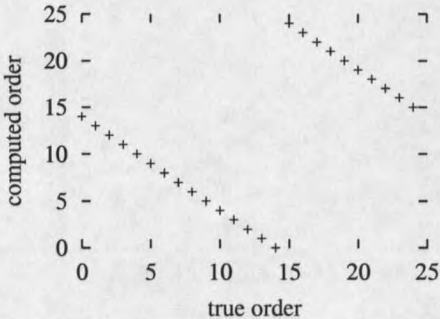
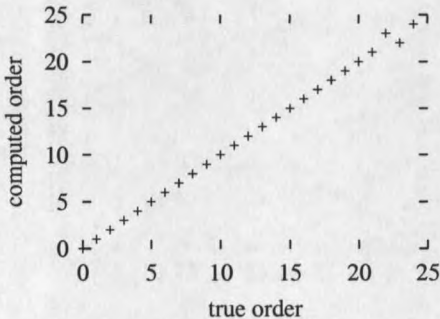
any section of the genome. Several clone coverage amounts were tested, the examples shown have average clone coverages of 5x and 3x. A clone coverage of 2x produces good results, but these results contained more disjoint orderings due to lack of overlap between clones in the orderings.

Table 1: Order and Overlap Results for 5x Coverage

Input Data	Order Results	Overlap Results
Data Set 1 Genome Length: 400,000 Number of Clones: 50 Clone Length: 40,000 Avg. Frag. Length: 450		False Positives: 0 False Negatives: 63
Data Set 3 Genome Length: 200,000 Number of Clones: 50 Clone Length: 20,000 Avg. Frag. Length: 350		False Positives: 0 False Negatives: 68
Data Set 5 Genome Length: 100,000 Number of Clones: 50 Clone Length: 10,000 Avg. Frag. Length: 400		False Positives: 0 False Negatives: 79

The ordering results were graphed on an X-Y axis with the y-axis representing the computed order, and the x-axis representing the true clone order. If the computed clone order matches the expected order, the clones should form a perfect diagonal line from the lower left hand corner to the upper right corner.

Table 2: Order and Overlap Results for 3x Coverage

Input Data	Order Results	Overlap Results
Data Set 2 Genome Length: 400,000 Number of Clones: 25 Clone Length: 50,000 Avg. Frag. Length: 450		False Positives: 1 False Negatives: 16
Data Set 4 Genome Length: 200,000 Number of Clones: 25 Clone Length: 25,000 Avg. Frag. Length: 350		False Positives: 0 False Negatives: 11
Data Set 6 Genome Length: 100,000 Number of Clones: 25 Clone Length: 12,000 Avg. Frag. Length: 400		False Positives: 0 False Negatives: 16

The overlap results are shown as the number of computed false positive and false negative overlaps between clones. A false positive occurs when a clone pair is said to be overlapping when there should be no overlap between the clones. A false negative occurs when a clone pair is said to be disjoint when they should be overlapping. Since the strictness level can be adjusted by the user, the overlap strictness level can be lowered to decrease the number of false negatives at the risk of increasing the false positives. False positives are considered worse than false negatives because they suggest that a researcher can characterize less clones for the desired final coverage when in actuality the clones will not be overlapping. Table 1 shows the overlap results for clone orderings with 5x coverage and Table 2 gives the overlap results obtained for clone orderings with 3x coverage. The number of false positives appears to increase as the number of clones and the clone coverage increases. Decreasing the strictness level a small amount for these data sets dramatically decreases the false negatives, but also introduces false positives.

The ordering results for each data set shown in Table 1 were nearly perfect. In each case, the sections that were not perfectly ordered can easily be explained by looking at the fingerprint data. In Data Set 1, there was one reversal where clones 16 and 17 were in reversed order. These two clones were identical except for the very first and last fragments, meaning only the starting and ending positions were slightly different. This is also the case with the reversals seen in Data Set 3 and Data Set 5. Data Set 1 also shows clones 48 and 49 separated from the rest of the clones in the order. Looking at the fingerprint data, these two clones overlap with each other, but neither clone overlaps with any other clone in the ordering, therefore it is expected that they would be disjoint. The entire orderings for Data Set 3 and Data Set 5 are reversed. Since there are no set clones for the starting or ending positions, this can

not be prevented with this algorithm.

The ordering results for the clones with 3x coverage, shown in Table 2, are also nearly perfect. The ordering for Data Set 2 is correct but the entire genome is reversed. The orderings in Data Set 4 are also completely reversed. Data Set 6 has one clone reversal where clones 22 and 23 are switched. These clones are nearly identical except the very beginning and the end of the clones, therefore, it would be expected that either clone could fall before the other in the final ordering. Data Set 4 shows two disjoint orderings. The first ordering contains clones zero through fourteen in perfect order, and the second ordering contains clones fifteen through twenty-four in their correct order. Looking at the fingerprint data generated, it was clear that there was no overlap between clones 14 and 15, and no other clones between the two orderings overlapped with each other. It is possible that the lower coverage in this data set allowed too few clones to fall within this section of the genome.

Running Time

Time calculations were done on a number of real and synthetic data sets in order to get an idea of the typical running time the user would encounter. All of these tests were run using system clock time, which does not take into account any system induced waits that may have occurred. Several time trials were run for each data set, the average time is shown rounded to the nearest millisecond (Table 3 and Table 4). All times were fairly similar, with no outliers for any of the time trials.

Calculations done on a data set take the longest on the first time because many variables are instantiated, and arrays are set-up. Subsequent calculations, with the original information imported, often take about half as long to complete. The times used here are all from first calculations on the data. The amount of memory allocated

to the JRE does not significantly affect the running times for the data sets tested. It is likely that none of the test data sets alone reached the maximum memory allocated even when it was as low as 16MB. These tests were run singly, re-opening CloneOrder for each test. There was a notable difference in running time when using the IBM Java Runtime Environment compared with the one created by Sun Microsystems; the Sun Microsystems JRE was considerably faster. Tables 3 and 4 show running times tested on a 500 Mhz computer running Red Hat Linux 7.0 with 128 MB memory allocated to the Sun Microsystems JRE.

Table 3: Synthetic Data Time Trials

Data Set	Genome Length (base pairs)	Number of Clones	Calculate Probabilities (milliseconds)	Calculate Order (milliseconds)
T1	400,000	50	798	2222
T2	400,000	25	639	704
T3	200,000	50	390	644
T4	200,000	25	339	683
T5	100,000	50	783	2278
T6	100,000	25	329	682

Table 4: Real Data Time Trials

Data Set	Number of Clones	Calculate Probabilities (milliseconds)	Calculate Order (milliseconds)
R1	31	560	1009
R2	25	500	685
R3	18	226	310
R4	14	184	275
R5	9	133	144

CHAPTER 7

CONCLUSION

The increase in automation of the techniques used to acquire data in the biological science field has led to an overwhelming amount of data to interpret by hand. The CloneOrder software described in this thesis will aid in the physical mapping of genomes by ordering and determining the overlap of clones based on their AFLP fingerprint data. The algorithm in this software orders the clones based on the fingerprint similarity between clones. Clones that share similar fingerprint patterns are likely to have been derived from the same segment of the original genome. This information is used to create an overlap value matrix that shows the confidence of overlap between any two clones. Permutations of this matrix are scored, and the final ordering is determined by this score. The overlap value matrix and the ordering of most data sets can be computed in under two seconds on a 500 Mhz computer. CloneOrder returns a graphical display of the clone order and the overlap between clones. This information can then be used to choose specific clones for further characterization or genome sequencing.

Future Work

The CloneOrder software was created specifically for the data created by the Plant Molecular Genetics department at Montana State University. Although many laboratories may use the same data format, not all laboratories would have the same input information. A common technique used to retrieve sizing data is to extract the data from scanned gel images of the BAC's restriction digests. Although this technique is very laborious, and does not give precise sizing information, it may be

valuable for CloneOrder to allow input from this type of data so the software will be accessible to a greater number of researchers.

Another option that may be useful to include in this software would be some editing functions. If a researcher determines that one or more clones in the contig should be removed, this can currently only be done by removing the clone, and re-importing the data. Editing functions will allow the user to remove questionable clones, and re-order the remaining clones.

BIBLIOGRAPHY

- [1] Benham, James. Genographer. Software.
http://hordeum.oscs.montana.edu/genographer.
- [2] Diekhoff, George. Statistics for the Social and Behavioral Sciences: Univariate, Bivariate, Multivariate. Dubuque: Wm. C. Brown Publishers, 1992.
- [3] Fasulo, Daniel. Algorithms for DNA Restriction Mapping. Ph.D. Dissertation. University of Washington, 2000.
- [4] Gilchrist, Warren G. Statistical Modelling with Quantile Functions. Boca Raton: Chapman & Hall, 2000.
- [5] Gusfield, Dan. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge: The Press Syndicate of the University of Cambridge, 1997.
- [6] Horowitz, Ellis, and Sartaj Sahni. Fundamentals of Computer Algorithms. Rockville: Computer Science Press, Inc, 1984.
- [7] Kachigan, Sam Kash. Statistical Analysis An Interdisciplinary Introduction to Univariate & Multivariate Methods. New York: Radius Press, 1986.
- [8] Kanazin, Vladimir. Assistant Research Professor, Plant Genetics Department, Montana State University, personal communication.
- [9] Keppel, Geoffrey. Design and Analysis A Researcher's Handbook. Third Edition. Englewood Cliffs: Prentice-Hall Inc., 1991.
- [10] Moore, David S. The Basic Practice of Statistics. 2nd Edition. New York: W.H. Freeman and Company, 2000.
- [11] Montgomery, Douglas C., and George C. Runger. Applied Statistics and Probability for Engineers. 2nd Edition. New York: John Wiley & Sons, Inc., 1999.
- [12] Mumey, Brendan. A Powerful Clone Overlap Test for Restriction Fragment Mapping. Unpublished manuscript, 1996.
- [13] Mumey, Brendan. Some Computational Problems in Genomic Mapping. Ph.D. Dissertation. University of Washington, 1997.
- [14] Sahni, Sartaj. Data Structures, Algorithms, and Applications in C++. Boston: The McGraw Hill Companies, Inc, 1998.

- [15] Theodoridis, Sergios and Konstantinos Koutroumbas. Pattern Recognition. San Diego: Academic Press, 1999.
- [16] Vos, Pieter, Rene Hogers, Marjo Bleeker, Martin Reijans, Theo van de Lee, Miranda Hornes, Adrie Frijters, Jerina Pot, Johan Peleman, Martin Kuiper, and Marc Zabeau. "AFLP: a new technique for DNA fingerprinting." Nucleic Acids Research, 23(21):4407-4414, 1995.
- [17] Walker, J.M., and E.B. Gingold. Molecular Biology and Biotechnology. 3rd Edition. Cambridge: The Royal Society of Chemistry, 1993.

MONTANA STATE UNIVERSITY - BOZEMAN



3 1762 10373521 1