INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Some Computational Problems from Genomic Mapping

by

Brendan Marshall Mumey

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

Doctor of Philosophy

University of Washington

1997

Approved by Walter L. Ruzzo
(Chairperson of Supervisory Committee)

Richard M. Karp

Richard Anderson

Program Authorized
to Offer Degree Computer Science and Engineering

Date 23 July 1997

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to University Microfilms, 1490 Eisenhower Place, P.O. Box 975, Ann Arbor, MI 48106, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature 

Date 22 - July - 1997

University of Washington

Abstract

Some Computational Problems from Genomic Mapping

by Brendan Marshall Mumey

Chairperson of Supervisory Committee: Professor Walter L. Ruzzo
Computer Science and Engineering

Modern molecular biology has presented computer science a host of interesting theoretical problems to work on. This thesis presents new algorithms for two computational problems from genomic mapping: The first problem arises in the context of building a restiction fragment map of the genome. The main technical result is a quickly-computed estimate of the *a posteriori* probability that a pair of clones overlap given their restriction fragment fingerprints. The computed pairwise overlap probabilities are used to search for a locally most-likely clone ordering. A correct or nearly correct clone ordering is needed for the subsequent mapmaking step of fragment identification. The second problem is to find the locations of probes along a genome given separation distance information for certain pairs of probes. The model considered is quite general: The input consists of a collection of probe pairs and an interval for the genomic distance separating each pair. We present an algorithm to determine the set of feasible probe positions up to global reversal and translations. Because the distance intervals are only known with some confidence level, some may be erroneous and must be removed in order to find a consistent map. A novel randomized technique for detecting and removing bad distance intervals is described.

TABLE OF CONTENTS

List of Figures iii

List of Tables iv

Chapter 1: Introduction 1

1.1 Genomic Mapping 2

1.2 Physical Mapping 3

1.3 Contributions of this Thesis 5

Chapter 2: Finding Clone Overlaps and Orderings from Restriction Fragment Fingerprints 8

2.1 Introduction 8

2.2 A Stochastic Model of Restriction Fragment Fingerprinting 11

2.3 The *A Posteriori* Probability of Overlap 12

2.4 Finding Clone Orderings and Interleavings 15

2.5 Experimental Results 19

2.6 Discussion and Future Work 26

2.7 Calculating the Chance of Hitting a Target 26

Chapter 3: A Fast Heuristic Algorithm for a Probe Mapping Problem 32

3.1 Introduction 32

3.2 The Problem 33

3.3 Finding Feasible Edge Orientations 37
3.4 Detecting Errors 39
3.5 Experimental Results 43
3.6 Future Work 47

Chapter 4: Conclusions **50**

Bibliography **51**

LIST OF FIGURES

2.1 Restricton fragment fingerprinting 9
2.2 Comparison of overlap test performance 20
2.3 Ordering results for data set g0771 24
2.4 Ordering results for data set g1862 25
2.5 Ordering results for data set g1862+g1346+g0771 25
2.6 Fragments lengths are "tossed" into the target bins 27

3.1 Definition of the probe location problem 34
3.2 Representing probe position solutions 35
3.3 Each individual 4-cycles is feasibly orientable, but they cannot be simultaneously oriented. The left cycle requires that the middle edges run in the same direction, while the right cycle requires the middle edge run in opposite directions. 39
3.4 A suspect-identification instance for which the mob-recognition criteria does not hold. The black points represent bad elements and the gangs present are circled. 42
3.5 DNA folding along a random walk backbone 48

LIST OF TABLES

2.1 Length measurement errors 21

2.2 Synthetic data results (25 clones) 22

2.3 Synthetic data results (50 clones) 23

2.4 Results of accuracy check 31

3.1 Experimental results for error-free data sets (40 probes) 45

3.2 Experimental results for data sets with errors (40 probes, 200 edges) . 46

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my advisor, Dr. Larry Ruzzo and to the other faculty members I have interacted with during my graduate study. I would also like to acknowledge my fellow theory students for numerous stimulating interactions. Finally, I would like to thank my family for their constant love and my friends who have made my stay in Seattle a very pleasant experience.

Brendan Mumey
April, 1997

Chapter 1

INTRODUCTION

Modern molecular biology is increasingly becoming an information science. It has supplied a number of challenging computational problems in the last twenty years. These range from sequence comparison and alignment problems to phylogeny construction to protein and RNA structure prediction to genomic mapping and sequence assembly to gene-finding. Recent advances in DNA array technology offer a new problem arena in gene expression analysis.

There are several aspects to computational biology problems that make them interesting targets for computer science research. They typically have a combinatorial flavor for which algorithmic sophistication is necessary. Often the data sets are large, so solutions must be resource efficient. Real data may contain errors that must be detected and dealt with. Previous work in computational biology has required non-trivial algorithms and data structures including: suffix trees, approximation schemes, hidden Markov models, simulated annealing and local search heuristics.

Some computational biology problems also have a machine-learning flavor. Cellular behavior appears determined by the rules encoded in DNA, the current biochemical "state" the cell is in, and the external environment. Information flowing into the cell (in the form of signalling factors), is processed by the cell in different ways. Depending on the information, the cell may respond by increasing the production of a particular protein, entering into a new phase of development, etc. Since the mechanisms of the cell are not directly observable, a operational model must be pre-

dicted based on observable behavior and the known constraints of cell biochemistry. Predicting the model can be viewed as a very general computational learning problem. Since there is an experimental cost to making observations, learning as much as possible with the fewest number of observations is desirable.

This thesis presents new algorithms for two compuational problems in genomic mapping. We begin with some background on genomic mapping and introduce the specific problems at the end of this chapter. Details will follow in the subsequent two chapters.

1.1 Genomic Mapping

All biological life is based on the developmental and functional instructions encoded in genomic DNA. DNA is a long chain molecule composed of four base nucleotides (bases), denoted A, C, T and G. It can be single or double stranded. Double stranded DNA takes the form of a double helix and a base on one strand must complement the opposing base on the other strand. Base A complements T and base C complements G. Cells in eukayotic organisms (all higher animals and plants) contain a nucleus in which the cell's *genome* resides. The genome provides a "blueprint" for the function of the cell. It consists of a set of *chromosomes*, each of which contains a long chain of double stranded DNA. The fundamental constituent unit of genomic DNA is the *gene*. Genes are subsequences that provide specific instructions to the cell for manufacturing proteins. Cell behavior is determined by the types and quantities of the proteins the cell produces. A central problem of molecular biology is to map the structure of the genome. There are two basic types of genomic maps: *Physical maps* supply information about the actual DNA sequence composing the genome. *Genetic maps* specify information about the relative location of markers (genes and polymorphisms) within the genome. A *polymorphism* is a detectable sequence variation within a population. Genetic maps are made by examining the frequency that a pair

of markers are coinherited. This gives a rough indication of the physical distance, in terms of the number of bases, separating the markers. *Physical maps* are made from assays on actual DNA samples from the organism being mapped. They provide more detailed information into the structure of a chromosome. As this thesis focuses on two computational problems from physical mapping we begin with an overview of the subject.

1.2 Physical Mapping

A physical map of a chromosome provides information about the relative order and position of *landmarks* along the genome. Landmarks are typically short segments of DNA, ranging from five to several thousand bases in length. There are two basic types: *probes* and *restriction sites.* Probes are sequences that hybridize (stick) to complementary DNA. Several laboratory techniques can be used to detect probe hybridization. A *restriction enzyme* cleaves DNA at a specific restriction site within the genome. These sites are short (usually 6-10 base) subsequences. For example, the enzyme *Eco*RI cuts at the subsequence GAATTC. Physical maps complement genetic maps in finding genes responsible for observable traits. Genetic maps provide a rough location of the gene in question (for example, between two polymorphic markers). A physical map supplying more detail between these two polymorphic markers can then be used to narrow in on the gene's location. Physical maps are typically made with *clone libraries*, although the second result in this thesis addresses a physical mapping approach that does not require a clone library. Clone libraries are overlapping collections of longer genomic DNA fragments, anywhere from 20 Kb (kilobases) to several Mb (megabases), that have been isolated and copied using recombinant DNA techniques. Maps made with clones libraries can be used to find *minimal clone tilings.* A minimal clone tiling a minimal set of clones that can be ordered so that consecutive clones in the order overlap and each location in the

genomic region being mapped is contained within a clone in the set. More-detailed mapping and sequencing efforts need only be performed on the clones in a minimal clone tiling. More background on probe mapping and restriction fragment mapping follows.

1.2.1 Probe Maps

There are several approaches to making physical maps with a set of probes. Most make use of a clone library and test which probes hybridize to each clone. The exceptions are the radiation hybrid and fluorescent in-situ hybridization methods that yield probe separation information directly. Longer probes such as *sequence tagged sites* (STSs) occur uniquely within the genome. Thus, the co-occurence of an STS in two clones is an unambiguous indication that they overlap. The Whitehead Institute's STS-content map of the human genome reported in [14] was made with STSs. Shorter probes occur multiple times within the genome. The set of probes that a clone contains is called its *fingerprint.* In [4], Alizadeh *et al.* present an algorithm for physical mapping with non-unique probes. They derive a computable estimate of the likelihood of a particular clone arrangement given the clone fingerprints. Heuristic search is then used to find a locally most-likely clone arrangement. The resulting arrangement indicates the approximate locations of each probe within the genome and their separations. An issue that all methods must cope with is experimental error. Clone fingerprints may contain both false positive and false negative probe hybridizations. Another type of error is a result of *chimeric clones.* These are clones that are not contiguous segments of genomic DNA, but are the unions of several disjoint segments that were accidentally fused together during the cloning process. Alizadeh *at al.* [2] and Istrail and Greenberg [13, 15] separately propose algorithms to construct physical maps from possibly chimeric clone libraries that have been fingerprinted with unique probes. Their methods also cope with some errors in the probe hybridization data. The problem of how to make physical maps from chimeric

clone libraries fingerprinted with non-unique probes is open at this time.

1.2.2 Restriction Fragment Maps

Restriction fragment maps are made with clone libraries. Instead of probes, a set of restriction enzymes is used to fingerprint the clones. Restriction sites occur more frequently than probes. In a random DNA string where each of the four bases is equally likely, one would expect an *Eco*RI restriction site every $4^6 = 4096$ bases. Although real DNA is obviously not random, restriction sites do appear to be distributed more or less randomly. Each restriction enzyme is applied to a representative copy of every clone. The clones are cleaved at the restriction sites they contain into a number of fragments. The lengths of these fragments are measured with a process called gel electrophoresis. The fingerprint of a clone is a tuple of components, one for each enzyme used. The component for a particular enzyme is the multiset of fragmet lengths produced when the clone is digested with that enzyme. If two clones share a number of similar-length fragments in each component, then this is evidence that they overlap. Building a restriction fragment map involves assembling the clones into the correct order and identifying which restriction fragments are shared by which clones. One tool currently in use for restriction fragment mapping is Gillet *et al.*'s DNAM program [12]. DNAM currently requires a lot of user interaction, so another tool called RMAP is being developed by Fasulo *et al.* [9] in hopes of producing accurate maps with less human effort required. RMAP relies on a heuristic algorithm to identifying shared fragments given the correct clone ordering. One of the research contributions of this thesis is a method for ordering clones.

1.3 Contributions of this Thesis

The main research contributions of this thesis are new algorithms for two problems from genomic mapping.

1.3.1 Ordering Clones From Restriction Fragment Fingerprints

We address the problem of inferring the underlying ordering and overlap relationship for a set of clones based on restriction fragment fingerprints of each clone. The main technical result is a quickly-computed estimate of the *a posteriori* probability that two clones overlap given their observed fingerprints. This calculation depends on a novel estimate of the probability of winning a certain occupancy game. Overlap probabilities are computed for all pairs of clones. Based on these pairwise probabilities, an estimate is derived for the overall likelihood of a proposed clone ordering. Search heuristics are used to find a locally most-likely clone ordering. A correct or nearly correct clone ordering is needed for the subsequent mapmaking step of fragment identification.

1.3.2 Mapping Probes with Noisy Pairwise Distance Data

A new heuristic algorithm is presented for mapping probes to locations along the genome, given noisy pairwise distance data as input. The model considered is quite general: The input consists of a collection of probe pairs and an interval for the genomic distance separating each pair. Because the distance intervals are only known with some confidence level, some may be erroneous and must be removed in order to find a consistent map. A novel randomized technique for detecting and removing bad distance intervals is described. The technique could be useful in other contexts where partially erroneous data is inconsistent with the remaining data. These algorithms were motivated by the goal of making probe maps with inter-probe distance confidence intervals estimated from fluorescence in-situ hybridization (FISH) experiments. Experimentation was done on synthetic data sets (with and without errors) and FISH data from a region of human chromosome 4. Problems with up to 100 probes could be solved in several minutes on a fast workstation. In addition to FISH mapping, we describe some other possible applications that fall within the problem model. These

include: mapping a backbone structure in folded DNA, finding consensus maps between independent maps covering the same genomic region, and ordering clones in a clone library.

Chapter 2

FINDING CLONE OVERLAPS AND ORDERINGS FROM RESTRICTION FRAGMENT FINGERPRINTS

2.1 Introduction

This chapter addresses the problem of inferring the underlying ordering and overlap relationship for a set of clones based on restriction fragment fingerprints of each clone. The main technical result is a quickly-computed estimate of the *a posteriori* probability that two clones overlap, given their observed fingerprints. This calculation depends on a estimate of the probability of winning a certain occupancy game. Overlap probabilities are computed for all pairs of clones. Based on these pairwise probabilities, an estimate is derived for the overall likelihood of a proposed clone ordering. Search heuristics are used to find a locally most-likely clone ordering. A correct or nearly correct clone ordering is needed for the subsequent mapmaking step of fragment identification. We begin with some background on the problem. The main results will follow in subsequent sections.

A *physical genomic map* specifies information about the order and physical separation of landmarks along the genome. One approach to physical mapping is restriction fragment mapping. Restriction fragment maps are made using a set of of *clones* called a clone library and a set of *restriction enzymes*. The clone library consists of overlapping segments copied (cloned) from the genomic region being mapped. A restriction enzyme cleaves a DNA sequence at specific restriction sites. These sites are short substrings of DNA specific to the enzyme used. For example, the enzyme *Eco*RI cuts at the substring GAATTC. Thus, in a random DNA string where each of the four

bases is equally likely, one would expect an *Eco*RI restriction site every $4^6 = 4096$ bases. Although real DNA is obviously not random, restriction sites do appear to be distributed more or less randomly.

Restriction enzymes are used to *fingerprint* clones. Each restriction enzyme is applied to a representative copy of every clone. The clone will be cleaved into a number of fragments at the restriction sites it contains. The lengths of these fragments are measured by gel electrophoresis. The multiset of lengths produced for each particular enzyme forms one component of the *clone fingerprint.* Figure 2.1 presents an overview of the process. If two clones have a number of similar fragment lengths

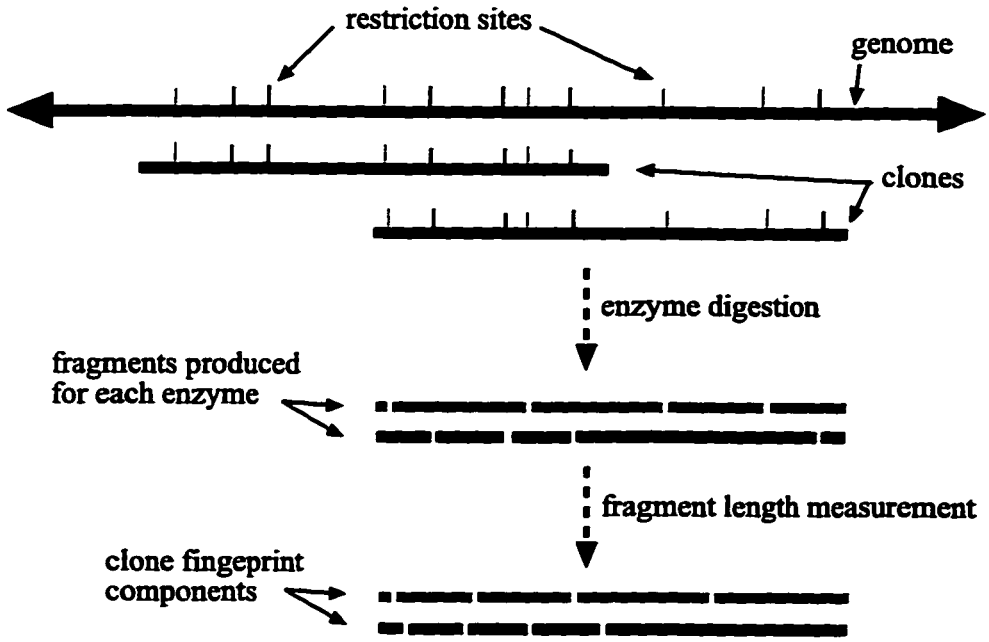


Figure 2.1: Restricton fragment fingerprinting

in each component of their fingerprints, then this suggests that they overlap. We present a new estimate of the *a posteriori* probability that two clones overlap given their measured fingerprints. Posterior overlap probabilities are computed for all pairs of the clones. These pairwise probabilities are used to estimate the likelihood of a

proposed ordering of the clones. This is done by estimating the likelihood of the most likely interleaving of the clone endpoints that is compatible with the specified clone ordering. Local search heuristics and simulated annealing are used to find a locally optimal clone ordering. The clone ordering produced can then be used as input to a fragment identification package such as the RMAP system [9].

Previous attempts at estimating the overlap probabilities given restriction fragment fingerprints have been made. In [12], Gillet *et al.* describe a heuristic *measured overlap statistic* (MOS) score that is correlated with overlap probability. In [6], Balding and Torney give a tight mathematical estimate of pairwise overlap probability, but it appears computationally very expensive to compute because a sum is taken over all possible permutations of the restriction fragments in each of the clones in consideration. The authors state that only a few terms contribute significantly to the sum but do not suggests a means to determine these terms. In [22], Nelson and Speed develop overlap likelihood estimates based on the probability of observing a particular vector for each pair of clones considered. The vector has dimension equal to the number of possible fragment lengths. Each position in the vector is one of four values indicating whether a fragment of the corresponding length occurred in each of the clones. The stochastic model they use assumes that the dimensions in the vector are independent. This is not true for clone libraries with similarly-sized clones. The most general estimate seems to be fairly computationally intensive, requiring an expectation-maximization step to first estimate a parameter in the model and a nontrivial numerical integration for each pairwise likelihood calculation. In contrast, the computational time of the method presented in §2.3 is exponential in the number of fragments whose lengths are deemed similar enough to match. In practice, the calculation is quite fast (20 matching fragments takes about a second on a Sun Sparc20 workstation).

This chapter is organized as follows: The next section presents a stochastic model for the distribution of clones in the genome and the generation of their fingerprints.

In §2.3, we derive an estimate for the *a posteriori* probability that two clones overlap based on matching fragments in their fingerprints. An estimate of the probability of winning a particular occupancy game is needed. The analysis is somewhat technical so we defer it to section §2.7 to ease reading the main results. In §2.4, we present a clone order evaluation function and describe the local search techniques used to find a good order. Experimental results on synthetic and real data are given in §2.5.

2.2 A Stochastic Model of Restriction Fragment Fingerprinting

The following model of genome fingerprinting is based on the model proposed by Arratia *et al.* [5]. A continuous model was chosen for generality. For typical clone sizes and enzyme rates, it should be indistinguishable from a discrete model. There are several model parameters:

- G - the length of the genome (a real number)
- n - the number of clones in the library
- $\{l_i\}_{i=1}^{n}$ - l_i is the length of clone C_i
- m - the number of restriction enzymes
- $\{\lambda_j\}_{j=1}^{m}$ - λ_j is the *cutting rate* of enzyme E_j

In point form, the stochastic model is as follows:

1. Each clone is a closed interval contained in the real interval $[0, G]$.
2. For a clone of length l, the left-hand end point is distributed independently and uniformly in the interval $[0, G - l]$;
3. For $j = 1, 2, \ldots, m$, the cut sites of enzyme E_j are distributed in $[0, G]$ according to a Poisson process of rate λ_j. These m Poisson processes are independent.

Each enzyme cutting contributes a component to a clone's fingerprint. We formally define the *actual fingerprint* of clone C_i to be an m-tuple $F_i = \langle S_{ij} \rangle_{j=1}^{m}$, where S_{ij} is the multiset of lengths that clone C_i is partitioned into by the cut sites of enzyme E_j. We will refer to S_{ij} as the *j-th component of the actual fingerprint for clone* C_i. In practice, lengths are not measured exactly. We model this by assuming that for each measured length l there is some interval $k(l)$ in which the true length must lie. Furthermore, all lengths in $k(l)$ are assumed to be equally likely. Although this model does not account for variance, it simplifies the analysis and is thought to be a reasonable choice. Thus the input to the program are the *measured fingerprints* of each clone: $f_i = \langle s_{ij} \rangle_{j=1}^{m}$, where s_{ij} is the multiset of measured fragments of clone C_i when digested with enzyme E_j. We will further assume that the clone lengths are known; these can estimated from the measured fingerprints, if not known by other means.

2.3 The A Posteriori Probability of Overlap

This section describes an efficiently computed estimate $\tilde{p}_{ab}$ of the likelihood that clones C_a and C_b overlap given the similarly sized fragments in their fingerprints f_a and f_b. Let s_{aj} and s_{bj} be the measured fingerprint components of clones C_a and C_b for enzyme E_j. Consider the bipartite graph whose left vertices are the elements of s_{aj} and whose right vertices are the elements of s_{bj}. An edge connects length $x \in s_{aj}$ to length $y \in s_{bj}$ if and only if $k(x)$ intersects $k(y)$. This edge implies the possibility that the actual lengths corresponding to x and y are equal, as would be the case if the clones overlap and shared a fragment of that length. We find a maximum matching M_j in this graph. This matching is used to estimate the chance that the clones overlap. A greedy subroutine finds M_j in the following way. First, the vertices on each side of the graph are placed into two stacks in decreasing order. The top element from each stack is popped; if there is an edge between them, then this edge

is added to the matching, if not the largest is discarded and smallest is pushed back on the stack from which it came. This process is repeated until one of the stacks is empty and the current matching is output. It can be shown that this subroutine always finds a maximum-sized matching. The basic idea of the proof is to show that a maximum matching exists that contains the first edge selected by the subroutine and the result follows by induction on the size of the matching.

We define the *target* for matching M_j to be $T_j = \{\ k(x) \cup k(y)\ :\ (x,y) \in M_j\ \}$. A target is *hit* by a sequence of fragment lengths $\{x_i\}$ if there is a one-to-correspondence between the target intervals and fragment lengths such that $x_i \in I$, if x_i is the fragment length corresponding to interval I. Let E_{aj} be the event that T_j is hit by S_{aj} and let E_{bj} be the event that T_j is hit by S_{bj}. Let $\max_j$ be the event that the matching M_j was maximal, i.e. no other fingerprint lengths could be matched for enzyme j. The event that each M_j is a maximum matching is:

$$E = E_{a1} \wedge E_{b1} \wedge \max_1 \wedge \ldots \wedge E_{am} \wedge E_{bm} \wedge \max_m.$$

From the measured fingerprints, we know that the event E has occured. Let the unknown variable $\theta \in [0, \min(l_a, l_b)]$ indicate the amount that C_a and C_b overlap. Assume, without loss of generality, that $l_a \geq l_b$. By Bayes theorem, the *a posteriori* likelihood ratio r_{ab} of overlap is:

$$r_{ab} = \frac{P[\theta > 0|E]}{P[\theta = 0|E]} = \frac{P[\theta > 0]}{P[\theta = 0]} \cdot \frac{P[E|\theta > 0]}{P[E|\theta = 0]}$$

We consider each factor separately. As the clones are distributed uniformly at random,

$$P[\theta > 0] = \frac{1}{(G - l_a)(G - l_b)} \int_0^{G-l_a} \int_{\max(0,x-l_b)}^{\min(x+l_a,G-l_b)} dydx = \frac{(G - l_a - l_b)(l_a + l_b) + l_a l_b}{(G - l_a)(G - l_b)}.$$

Since $P[\theta = 0] = 1 - P[\theta > 0]$,

$$\frac{P[\theta > 0]}{P[\theta = 0]} = \frac{G(l_a + l_b) - l_a l_b - l_a^2 - l_b^2}{(G - l_a - l_b)^2}.$$

Given a fixed amount of clone overlap, the events that particular matchings occur for different enzyme digestions are independent. Thus

$$\begin{aligned}\frac{P[E|\theta>0]}{P[E|\theta=0]} &= \frac{\int_0^{l_b}\prod_{j=1}^m P[E_{aj}\wedge E_{bj}\wedge \max_j|\theta=x]\,\frac{dx}{l_b}}{\prod_{j=1}^m P[E_{aj}\wedge E_{bj}\wedge \max_j|\theta=0]}\\ &= \int_0^1 f(u)g(u)\,du\end{aligned}$$

where $x = l_b u$ and

$$\begin{aligned}f(u) &= \prod_{j=1}^m \frac{P[E_{aj}\wedge E_{bj}|\theta=l_b u]}{P[E_{aj}\wedge E_{bj}|\theta=0]}\\ &= \prod_{j=1}^m \frac{P[E_{aj}|E_{bj}\wedge\theta=l_b u]P[E_{bj}]}{P[E_{aj}|E_{bj}\wedge\theta=0]P[E_{bj}]}\\ &= \prod_{j=1}^m \frac{P[E_{aj}|E_{bj}\wedge\theta=l_b u]}{P[E_{aj}]}\end{aligned}$$

and

$$g(u) = \prod_{j=1}^m \frac{P[\max_j|E_{aj}\wedge E_{bj}\wedge\theta=l_b u]}{P[\max_j|E_{aj}\wedge E_{bj}\wedge\theta=0]}.$$

Clearly $f(0)=1$ and $g(0)=1$. When $u=1$, C_b is completely contained in C_a and so the event E_{bj} implies the event E_{aj}. Thus $f(1) = 1/\prod_{j=1}^m P[E_{aj}]$. We consider the numerator of the integrand in g when $u=1$: Let L_j equal the sum of the lefthand endpoints of the intervals in T_j and let R_j equal the sum of the righthand endpoints. The total lengths of the matched fragments is approximately $S_j = (L_j + R_j)/2$. If M_j is to be maximal, no additional cut sites can occur in the unmatched part of clone C_b (otherwise a new fragment would be shared between the clones and found in the matching). As the cut sites form a Poisson process, the chance that this occurs is $\int_{l_b-S_j}\lambda_j e^{-\lambda_j x}dx = e^{\lambda_j(l_b-S_j)}$. The event $\max_j$ says that no futher matches can be found. If C_a and C_b do not overlap and are of comparible size, the denominator should be roughly exponential in the normalized unmatched length of C_b, $\lambda_j(l_b - S_j)$. So $g(1) \approx e^{h\cdot\lambda_j(l_b-S_j)}$. A simple binomial model for the approximate shape of these functions indicates that they should be roughly the product

of a binomial density function and an exponential. For typical parameters, experimentation using Mathematica indicated that an exponential approximation of the form $f(u)g(u) \approx e^{cu}$ is reasonably acceptable for the purpose of evaluating the above integral. This model and further (somewhat ad hoc) experimentation suggests that $c = -\sum_{j=1}^{m}(\log P[E_{aj}] + 0.3\lambda_j(l_b - S_j))$ yields about the right range of values. We have

$$\frac{P[E|\theta > 0]}{P[E|\theta = 0]} \approx \int_0^1 e^{cu}\, du = \frac{e^c - 1}{c}.$$

The estimate of the likelihood ration r_{ab} is:

$$\tilde{r_{ab}} = \frac{G(l_a + l_b) - l_a l_b - l_a^2 - l_b^2}{(G - l_a - l_b)^2} \cdot \frac{e^c - 1}{c}. \tag{2.1}$$

We need to evaluate each $P[E_{aj}]$. The event E_{aj} can be viewed as the event of winning a certain occupancy game. For clarity of reading the main results, the discussion of this game is deferred to §2.7. Plugging these estimates into (2.1), gives a computable estimate of $\tilde{r_{ab}}$. Let p_{ab} be the probability that clones C_a and C_b overlap. Since $r_{ab} = \frac{p_{ab}}{1-p_{ab}}$, solving for p_{ab} yields

$$\tilde{p}_{ab} = \frac{\tilde{r}_{ab}}{1 + \tilde{r}_{ab}}. \tag{2.2}$$

2.4 *Finding Clone Orderings and Interleavings*

The pairwise overlap probability estimates found in the previous section are useful for evaluating clone orderings and interleavings. Listing the lefthand clone endpoints from left to right specifies a clone ordering. A clone *interleaving* indicates which clone pairs overlap. We will assume that no clone is properly contained within another. This is a somewhat unrealistic assumption, but simplifies the overlap relationship. An ordering is evaluated by finding the most likely interleaving that is compatible with it. Local search heuristics are used to find good orderings. We first describe the order evaluation method and then discuss the search heuristics.

2.4.1 Evaluating a Clone Ordering

Let π be a permutation. We wish to evaluate π as a possible clone ordering. An interleaving I of the clones is *compatible* with π if $(\pi(a), \pi(b)) \in I$ implies that $(\pi(i), \pi(j)) \in I$ for all $a \leq i, j \leq b$. The matrix representation of I consists of square blocks of ones centered along the main diagonal, when the rows and columns have been ordered by π. We make the approximation that the overlap probabilities (p_{ij}) between pairs of clones are independent. The likelihood of I is then,

$$L(I) \approx \prod_{(i,j)\in I} p_{\pi(i)\pi(j)} \prod_{(i,j)\notin I} (1 - p_{\pi(i)\pi(j)}).$$

Taking the negative logarithm yields a computable expression for the *cost* of an interleaving:

$$c_1(I) = -\sum_{(i,j)\in I} \log \tilde{p}_{\pi(i)\pi(j)} - \sum_{(i,j)\notin I} \log(1 - \tilde{p}_{\pi(i)\pi(j)}).$$

We are interested in finding an interleaving I^* compatible with π that minimizes c_1. One can be found by dynamic programming in $O(n^2)$ time. The idea is to note that an interleaving can represented as a path p above the main diagonal in the overlap matrix. The path follows the boundary of the region of ones in the overlap matrix. The path must have the property that, starting from position $(1,1)$ in the matrix, it moves either right or down and eventually reaches position (n,n). The reason for this is that if a one appears in position (i,j), the triangle with corners (i,i),(i,j) and (j,j) must contain only ones for the interleaving to be compatible with π. Let $h(j)$ be the lowest row index which the path goes through in column j. The cost of the path is the cost of the interleaving I_p associated with path. We have

$$c_1(I_p) = \sum_{j=1}^{n} a(j, h(j))$$

where

$$a(j,h) = \sum_{i=1}^{h-1} \log \tilde{p}_{\pi(i)\pi(j)} + \sum_{i=h}^{j} \log(1 - \tilde{p}_{\pi(i)\pi(j)}).$$

It is clear that all $a(j,h)$ can be computed in $O(n^2)$ time. Let $s(i,j)$ be the cost of an optimal partial path that ends in position (i,j). A partial path will be charged for the columns that it uses, i.e. $s(i,j) = \sum_{l=1}^{j} a(l,h(l))$. Since an optimal path through position (i,j) must extend an optimal path through position $(i-1,j)$,$(i,j-1)$, or $(i-1,j-1)$,

$$s(i,j) = \min(s(i-1,j), s(i,j-1) + a(j,i), s(i-1,j-1) + a(j,i)).$$

We can use dynamic programming to find $c_1(I^*) = s(n,n)$ in $O(n^2)$ time.

The cost function c_1 assumes that the pairwise overlap probabilities are independent. This is certainly not the case. If the clones are in the true order, then one would expect the sequence of $p_{i1}, \ldots, p_{in}$ to be close to zero for a while, then increase monotonically up to a peak at $p_{ii} = 1$ and then decrease monotonically down to near-zero values. We introduce another cost function c_2 to charge for the amount of variation in this sequence.

$$c_2(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n-1} |\max(\log \tilde{r}_{i(j+1)}, 0) - \max(\log \tilde{r}_{ij}, 0)|.$$

Log-likelihood ratios were used instead of probabilities to accentuate the variation for probabilities near 1. If the log-likelihood ratio was less than 0, then the value taken was 0. This should help remove misleading variation from non-overlapping pairs of clones. A good ordering should also have a low c_2 cost. We combine the two cost functions to get an overall cost function for a clone ordering that can be evaluated in $O(n^2)$ time:

$$c(\pi) = c_1(I^*) + 0.05c_2(\pi).$$

The c_2 weight factor of 0.05 was chosen so that each term was of similar size. Choosing lower or higher values had a negative effect on the quality of clone orderings produced.

2.4.2 Searching for Good Orderings

Simulated annealing and local search techniques are used to find a good clone ordering. The basic idea is to define a local neighborhood structure on permutations and to use this structure to search for an optimal one. A random neighbor of the current permutation is chosen. If it has a lower cost, then it becomes the current permutation. Simulated annealing is used to initially allow cost-increasing moves in hopes that they jump out of non-optimal local minima. After a certain number of moves only improving moves are accepted. After a fixed number of moves a deterministic valley-descending strategy is used. Each of the neighboring permutations are considered and the first cost-decreasing move found is taken. This is repeated until no cost-decreasing moves are found. The following moves define the neighbors of a permutation $\pi = \pi(1)\dots\pi(n)$.

- *2-opt*: Two indices $i < j$ are chosen as pivot points. The neighboring permutation is:

$$\pi' = \pi(1)\dots\pi(i-1)\pi(j)\pi(j-1)\dots\pi(i)\pi(j+1)\dots\pi(n).$$

- *swap*: Two indices $i < j$ are chosen to swap. The neighboring permutation is:

$$\pi' = \pi(1)\dots\pi(i-1)\pi(j)\pi(i+1)\dots\pi(j-1)\pi(i)\pi(j+1)\dots\pi(n).$$

- *rotate*: A rotation index i is chosen. The neighboring permutation is:

$$\pi' = \pi(n-i)\pi(n-i+1)\dots\pi(n)\pi(1)\pi(2)\dots\pi(n-i-1).$$

- *displace*: Two indices i and j are chosen. The neighboring permutation of the form:

$$\pi' = \pi(1)\dots\pi(i-1)\pi(i+1)\dots\pi(j)\pi(i)\pi(j+1)\dots\pi(n).$$

or

$$\pi' = \pi(1)\dots\pi(i)\pi(j)\pi(i+1)\dots\pi(j-1)\pi(j+1)\dots\pi(n).$$

2.5 Experimental Results

We begin with an experimental comparison of an overlap test based on the calculated posterior overlap probability versus the MOS score used by Gillet *et al.* [12]. Then the clone ordering/interleaving algorithm is tested on both synthetic and real data sets. In each case, the true map is known, so we can compare the computed clone order and interleaving against the true map. The real data sets were from the University of Washington Genome Center.

2.5.1 Comparison with the MOS statistic

For the problem of just deciding if a particular pair of clones overlap, we consider the following threshold-based test. For a threshold value τ, let

$$\mathrm{PRB}(C_a, C_b) = \begin{cases} 1 & \text{if } \bar{p}_{ab} \geq \tau \\ 0 & \text{if } \bar{p}_{ab} < \tau \end{cases}$$

For any fixed overlap detection test based on fingerprint similarity there will be some probability of false positives and negatives. If an overlap detection test minimizes the probability of a false negative for a bounded probability of a false positive, the test is said to be *most powerful.* The Neyman-Pearson lemma [10] of classical hypothesis testing shows that among tests of a particular hypothesis based on a certain observation, thresholding the *a posteriori* probability of the hypothesis given the observation yields the most powerful test. Thus, if the calculated estimate of the posterior overlap probability is close to correct, PRB should be close to optimal among all tests based on the knowledge of the fragment matchings and clones lengths. To get an idea of whether this is true in practice, we compare PRB against two versions of the MOS test, sumMOS and minMOS. (At this point, it is not clear to the author which one is used in the DNAM program.) Both tests are based on calculating a statistic x_j for each enzyme digest:

$$x_j = \frac{|M_j|^2}{|s_{aj}||s_{bj}|}.$$

The sumMOS overlap test is then defined as:

$$\text{sumMOS}(C_a, C_b) = \begin{cases} 1 & \text{if } \sum x_j \geq \tau \\ 0 & \text{if } \sum x_j < \tau \end{cases}$$

The minMOS is defined identically except the sum function is replaced with the min function. Figure 2.2 shows a plot for the false negative versus false positives for each of the three tests. The tests were performed on a synthetic clone library consisting of 20 clones, each of length 40,000 bases. The genome length was 200,000 bases and four enzymes were used, each with a cutting rate of 1/4096. It can be seen from the plots that PRB outperforms sumMOS by a factor of 20-40% and minMOS by a factor 30-50%, in the region of low false positives. This region is the most interesting to compare, as false positives tend to make it much harder to find correct clone orderings.

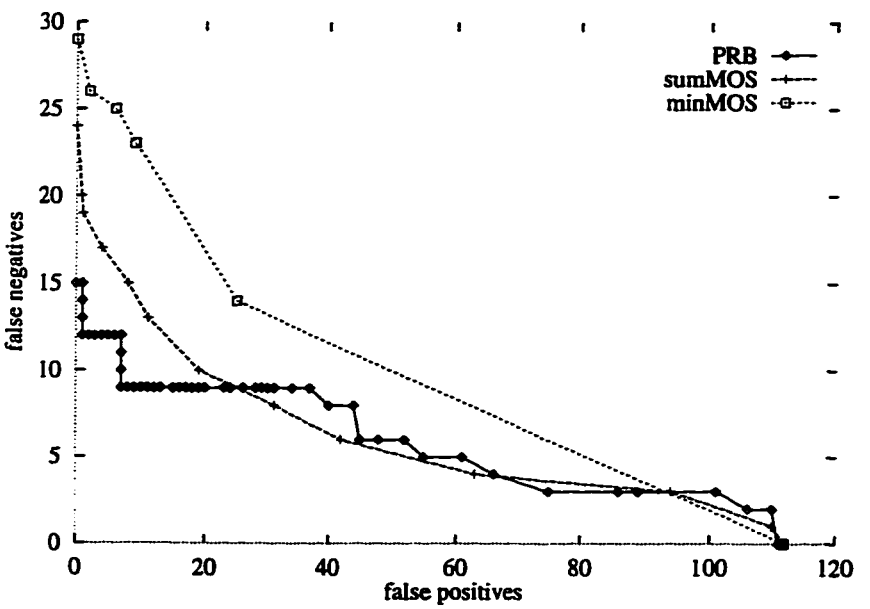


Figure 2.2: Comparison of overlap test performance

Table 2.1: Length measurement errors

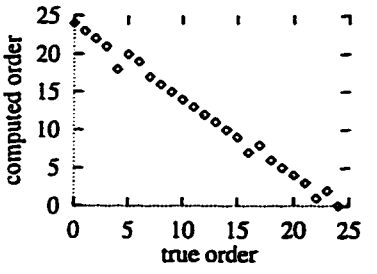
true length L	error value ϵ
< 317	0.04
317 - 562	0.03
563 - 1000	0.025
1001 - 1778	0.022
1779 - 3162	0.021
3163 - 5623	0.022
5624 - 10000	0.024
10001 - 13335	0.027
13336 - 17780	0.04
17781 - 23714	0.06
> 23714	0.10

2.5.2 Synthetic Data Tests

Table 2.2 shows the results of the clone ordering algorithm on two random clone libraries. In each case, there are 25 clones each of length 40,000 bases. They are sampled from a random genome of length 200,000 bases. This gives an average clone *coverage* of 5, where coverage is the expected number of clones containing a random location on the genome. In the first case, the clones were digested with three enzymes, each with cutting rate $1/4^6$. In the second case, five enzymes were used with the same cutting rate. For each clone fingerprint, a length measurement error was added to each length present. For a true length L, a measured length l was sampled uniformly from the interval $[L/(1+\epsilon), L(1+\epsilon)]$, where the error value ϵ depended on L according to Table 2.1 supplied by the University of Washington Genome center.

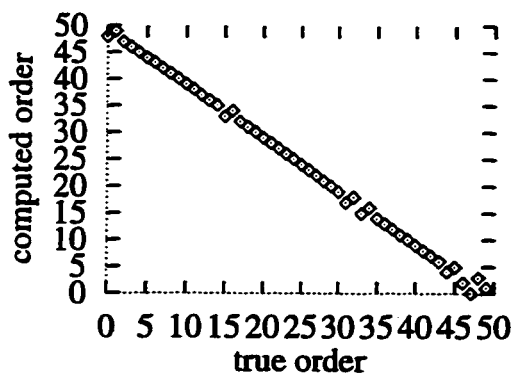
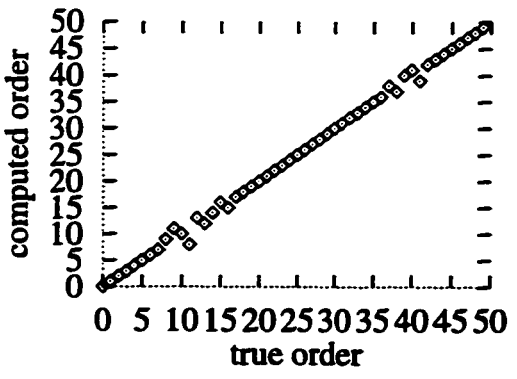
A *false positive* occurs if two clones are said to overlap when they are disjoint.

Table 2.2: Synthetic data results (25 clones)

m	false positives	false negatives	computer order vs. true order
3	0	38	
5	0	48	

Likewise, a *false negative* occurs if two clones are said to be disjoint when they do overlap. The table shows the number of false positives and negatives and a graph of the computed clone order versus the true clone order. Table 2.3 shows the results of two more experiments. The number of clones is now 50 and the clone length has been increased to 60,000 bases. The genome length has been increased to 300,000 (thus the coverage is 10). The number of enzymes used is varied between 3 and 5 as before and the cutting rate of each enzyme remains $1/4^6$. In all of the examples, the clone order found is close but not completely correct. The errors that do occur appear to be one of two types: local errors within contigs (runs of well-overlapped clones) where near-adjacent clones are out of order and errors in the global ordering

Table 2.3: Synthetic data results (50 clones)

m	false positives	false negatives	computer order vs. true order
3	0	119	
5	0	101	

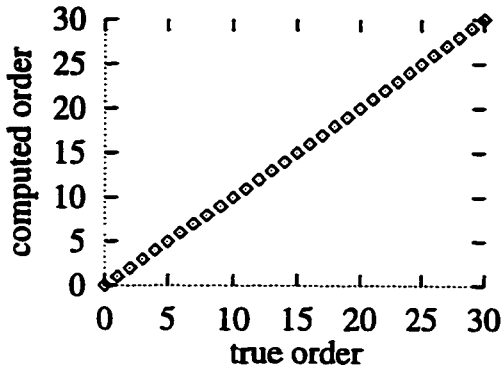


Figure 2.3: Ordering results for data set g0771

the contigs. The second type of error is expected from a uniform clone distribution; it is likely that there will be some areas of the genome that have low coverage. Finding a "bridge" of overlaping clones in these areas is difficult.

2.5.3 Real Data

Figures 2.3, 2.4, and 2.5 show the computed clone ordering versus the true ordering for several clone libraries that were fingerprinted using three restriction enzymes at the University of Washington Genome Center. The third library is a combination of clones from the first two libraries and another. These three libraries overlap. In each case, what is believed to be the correct map has been found by other methods. Thus, the true clone ordering is known. (Unfortunately, the overlap relationship is not supplied, so false positive and negative statistics for the computed overlap relationship cannot be stated.) The computed ordering is completely correct for the first two data sets and is off by two clone transpositions in the third.

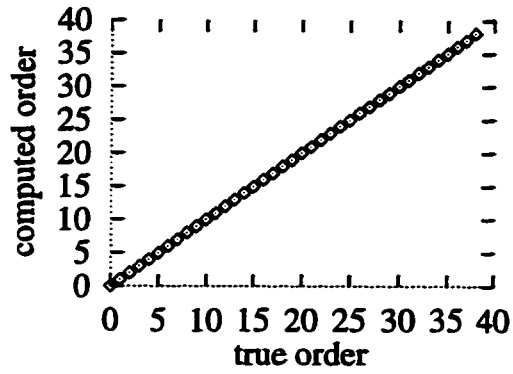


Figure 2.4: Ordering results for data set g1862

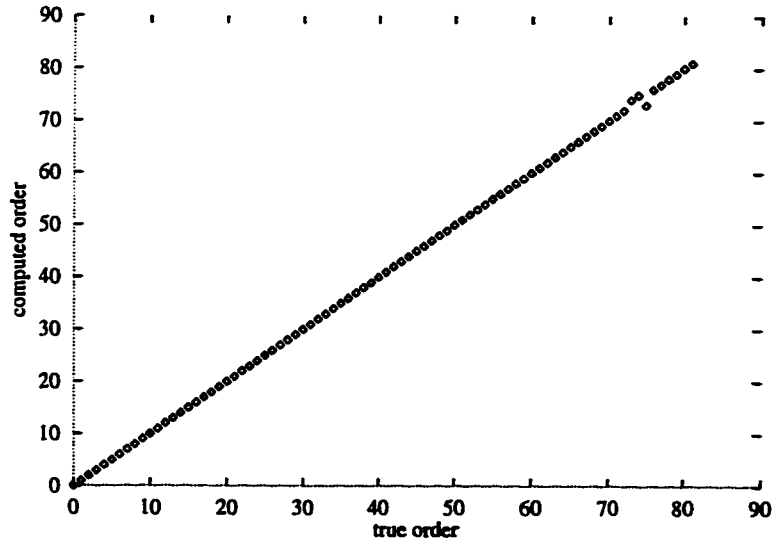


Figure 2.5: Ordering results for data set g1862+g1346+g0771

2.6 Discussion and Future Work

The clone-ordering algorithm appears fairly effective in finding a close to correct clone ordering. In all the cases where the true ordering was not found, it was because the computed ordering had a slightly lower score. This suggests that the scoring function might be improvable. It also suggests slightly sub-optimal solutions might be useful to look at. For example, one could find the best few sub-optimal orders in a small neighborhood of the optimal clone order. One possible neighborhood to consider would be all possible 2-opt moves consisting of two or three clones. This would add confidence in the parts of the optimal clone ordering that didn't change in the sub-optimal orders and possibly point out other parts in the optimal ordering that were more suspect.

In addition to the overlap decision test comparison, it would be interesting to see if the advantages of estimating the posterior overlap probability significantly affect the accuracy of the computed clone orderings. Some initial experimentation with just plugging the normalized sum-MOS score into the ordering/interleaving algorithm suggests that the estimated overlap probabilities do improve the accuracy of the computed clone ordering.

The time required to compute the overlap probability for a pair of clones grows exponentially with the number of matching fragments. In practice, this is not too much of a hindrance, as a probability calculation with 20 matching fragments can be done in about a second. For clone libraries of interest, 20 matching fragments is a definite indication that the clones overlap, so we limit all matchings to 20 fragments or less.

2.7 Calculating the Chance of Hitting a Target

We are interested in calculating the probability p_{hit} that a set of target intervals $T = \{[l_i, r_i]\}_{i=1}^{k}$ is hit by the fragments lengths produced by digesting a clone of

length s with an enzyme that cuts with rate λ. The distance d between consecutive points in a Poisson process of rate λ is distributed exponentially with parameter λ, i.e. $d \sim \lambda e^{-\lambda x}$. Starting from the left end of the clone, the lengths of consecutive fragments are repeated samples of this stribution. Samples are taken until the sum of the sampled lengths exceeds s, corresponding to the sampled cut site occuring past the right end of the clone. The target intervals can be viewed as bins along the real line and each fragment as a ball that is "tossed" to a position along the line according to the exponential distribution (see Figure 2.6). A ball landing in a bin simulates

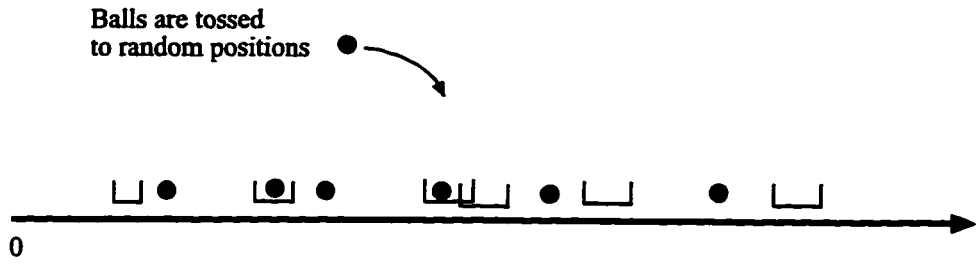


Figure 2.6: Fragments lengths are "tossed" into the target bins

a target interval being hit by a restriction fragment length. We will say that all the bins are filled if each bin can be put into one-to-one correspondence with a ball that has fallen into the bin. The probability p_G that all the bins are filled before the sum of the ball positions exceeds s will be a slight underestimate of p_{hit} as the rightmost fragment in the clone fingerprint is terminated by the clone end, not the next restriction site. There is a small probability p_X that this fragment will fill the last empty bin. Since $p_G \gg p_X$, we will just use p_G to estimate p_{hit}:

$$p_{\text{hit}} \approx p_G.$$

Let the sum of the ball positions (fragment lengths) be c. Let O_t be the event that the last unoccupied bin is filled on the t-th toss. Then

$$\begin{aligned} p_G &= \sum_{t \geq 0} P[c < s \wedge O_t] \\ &= \sum_{t \geq 0} P[c < s | O_t]\ P[O_t]. \end{aligned}$$

We first derive an estimate of $P[c < s|O_t]$. Since all the bins are occupied, one ball is conditioned to fall into each bin. Let the sum of these bin-filling ball positions be c_1 and the sum of the remaining ball positions be c_2. Let $L = \sum_{i=1}^{k} l_i$ and $R = \sum_{i=1}^{k} r_i$. The conditioning implies that $c_1 \in [L, R]$. Thus

$$P[c_2 \leq s - R] \leq P[c_1 + c_2 \leq s] \leq P[c_2 \leq s - L].$$

There is some conditioning on the remaining balls: Since the last bin is filled on the t-th move, the earlier balls are conditioned to avoid it. Since it seems difficult to do otherwise, we will accept the approximation that these balls are unconditioned. Thus, c_2 is the sum of $t - k$ independent exponential variates with parameter λ. The *Erlang* distribution $E_{\lambda,t-k}(x)$ describes this distribution. It follows that

$$E_{\lambda,t-k}(s - R) \leq P[c \leq s|O_t] \leq E_{\lambda,t-k}(s - L).$$

We will accept the approximation that

$$P[c \leq s|O_t] \approx E_{\lambda,t-k}(s - (L + R)/2).$$

A well-known relationship [19] between the Erlang and Poisson distributions shows that

$$E_{\lambda,t-k}(s - (L + R)/2) = e^{-\lambda(s-(L+R)/2)} \sum_{j=t-k}^{\infty} \frac{(\lambda(s - (L + R)/2)^j}{j!}.$$

From the previous discussion,

$$p_G = \sum_{t \geq 0} P[c < s|O_t]\ P[O_t]$$

$$\approx \sum_{t \geq k} e^{-\lambda(s-(L+R)/2)} \sum_{j \geq t-k} \frac{(\lambda(s-(L+R)/2))^j}{j!} P[O_t]$$
$$= e^{-\lambda(s-(L+R)/2)} \sum_{j \geq 0} \frac{(\lambda(s-(L+R)/2))^j}{j!} \sum_{t \leq j+k} P[O_t].$$

We next derive an estimate of $\sum_{t \leq j+k} P[O_t]$. This is the probablility that all the bins are occupied after $j+k$ ball tosses. Consider the following Markov process: The states in the process indicate which bins have been filled. If there are k bins, there are 2^k states. In practice, $k \leq 20$, so this is a tolerable space requirement. We label each state with a binary vector where a 0 in position i indicates that bin i is empty and a 1 indicates that it is occupied. For each state s there is a probability function $p_s(t)$ that indicates the probability of the process being in state s at time t. Thus $p_{00\ldots0}(0) = 1$ and $p_s(0) = 0$ for all $s \neq 00\ldots0$. For a state s, let $0(s)$ $(1(s))$ be the set of bins that are empty (occupied) in state s. For each $i \in 1(s)$, let $s - i$ be the state with the same empty bins as s, except that bin i is also empty. Let b_i be the probability that bin i is hit by a random ball toss, i.e. $b_i = \int_{l_i}^{r_i} \lambda e^{-\lambda x} dx$. The time dependence of the state probabilities is then given by the following expression:

$$p_s(t+1) = \sum_{i \in 1(s)} b_i \cdot p_{s-i}(t) + (1 - \sum_{i \in 0(s)} b_i) \cdot p_s(t).$$

In the case where the bins are pairwise disjoint, we claim that $\sum_{t \leq j+k} P[O_t] = p_{11\ldots1}(j+k)$. To see this, note that the probability that all bins are occupied after n moves is the sum over all permutations π that the bins are occupied in order π within n moves (because these events are pairwise disjoint). The probability that all bins are occupied in order π is the probability of following a path of vertices uniquely determined by π from state $00\ldots0$ to state $11\ldots1$. It is easy to show that the probability of being in state s at time t is the sum over all vertex paths from state $00\ldots0$ to state s of the probability of being in state s at time t by way of a particular vertex path. If a pair of bins intersect then the above argument doesn't hold: The chance of following a particular vertex path slightly underestimates the

chance of seeing the bins in the corresponding order as the probability of seeing the bins out of order is slightly decreased. For typical fingerprints, it is a relatively rare event for a fragment matching to have an intersecting pair of bins, so $p_{11\ldots1}(j+k)$ is a very good approximation to $\sum_{t\leq j+k} P[O_t]$ for our purposes.

If the state probabilities are represented in an array indexed by the binary state labels, the Markov process can stepped one time step by making a single backward pass through the array. (This is because all the dependencies go from lower indices to higher indices.) The estimate for p_{hit} is then

$$p_{\text{hit}} \approx e^{-\lambda(s-(L+R)/2)} \sum_{j\geq 0} \frac{(\lambda(s-(L+R)/2))^j}{j!} p_{11\ldots1}(j+k).$$

For $k = 20$, this sum can be evaluated in a few seconds. Some game simulations were done to check the accuracy of this estimate. On realistic cases, the estimate of p_{hit} was accurate within a few percent of the observed frequency. Table 2.4 shows the simulated frequencies versus predicted probabilities that three different target sets are hit by the fragment lengths of a segment of length 100,000 digested with an enzyme of rate 1/4096. The predicted value p was first computed and then $1000/p$ digestion simulations were performed and the frequency that the target was hit was reported. The target sets used were:

$$\begin{aligned} T_a &= \{[1000, 1200], [3000, 4000], [10000, 11000], [13000, 14000]\} \\ T_b &= \{[500, 550], [700, 800], [2000, 2200], [4600, 4800], [6000, 6300]\} \\ T_c &= \{[500, 530], [700, 740], [1000, 1100], [3000, 3200], [6600, 7000], [6800, 7300]\} \end{aligned}$$

Table 2.4: Results of accuracy check

target set	predicted probability	observed frequency
T_a	0.0284	0.0294
T_b	0.00837	0.00809
T_c	0.000751	0.000714

Chapter 3

A FAST HEURISTIC ALGORITHM FOR A PROBE MAPPING PROBLEM

3.1 Introduction

This chapter presents a new heuristic algorithm for mapping *probes* to locations along the genome given noisy pairwise distance data as input. The model considered is quite general: The input consists of a collection of probe pairs and a confidence interval for the genomic distance separating each pair. Because the distance intervals are only known with some confidence level, some may be erroneous and must be removed in order to find a consistent map. A novel randomized technique for detecting and removing bad distance intervals is described. The technique could be useful in other contexts where partially erroneous data are present. These algorithms were motivated by the goal of mapping probes with distance confidence intervals estimated from *fluorescence in-situ hybridization (FISH)* experiments[1]. As the problem is too big to solve by hand, some previous algorithm development have been tried, including: a seriation algorithm [7], a simulated annealing approach [23], and a branch and bound algorithm [24]. Due to their combinatorial nature, these methods are limited to problems with about 20 probes or fewer. The probe-location algorithm presented

[1] A FISH experiment measures the physical distance (on a microscope slide) between pairs of fluorescently marked probes hybridized to an interphase chromosome[28, 27]. For genomic distances of up to about 1-2 Mb, DNA folding can be described by a random walk model. Statistics can be used to estimate a confidence interval for the genomic distance (in base-pairs) between two probes given a measured sample of physical distances.

here has been implemented and can solve problems with 100 probes in several minutes on a fast workstation. Experimental results have been collected for synthetic data sets (with and without errors) and FISH data from a region of human chromosome 4.

We also suggest some other applications for which the probe-location algorithm could be useful: finding and mapping a *backbone* structure within folded DNA using FISH data, finding a *consensus map* from a set of independent maps covering a genomic region, and *ordering clones* in a clone library.

The remainder of the chapter is organized as follows: In the first section, the PROBE-LOCATION problem is formally defined and shown to be equivalent to a particular graph problem. Certain sparse instances of the problem are shown to be NP-complete. A heuristic algorithm to find all the feasible solutions to an instance of PROBE-LOCATION is described in the next section. Provided that the input is not from the small class of hard sparse instances, the algorithm is quite effective. The subsequent section presents a general randomized technique for finding and removing erroneous distance intervals in the input. Experimental results on synthetic data sets as well as FISH distance data for human chromosome 4 are given in the next section. Other problems for which the probe-location algorithm could be useful are described in the last section. Possible generalizations to the error model are also briefly mentioned.

3.2 The Problem

A formal definition of the probe-location problem is given in Figure 3.1. In the remainder of this section we show that the probe-location problem is equivalent to finding feasible edge orientations in an edge orientation graph and show that finding such orientations can be NP-hard in certain cases.

If $(x_1, \ldots, x_n)$ is a feasible solution, then for each separation constraint (i, j, l, u),

PROBE-LOCATION

Input: A list of probes $\{p_1, \ldots, p_n\}$ and a list of m separation contraints of the form $\{(i, j, l, u))\}$, with $0 \leq l \leq u$ in all cases. A separation constraint (i, j, l, u) indicates that that the distance between p_i and p_j falls in the interval $[l, u]$.

Output: A description of the set of feasible probe positions $\{(x_1, \ldots, x_n)\}$ such that $|x_i - x_j| \in [l, u]$ for all separation contraints (i, j, l, u).

Figure 3.1: Definition of the probe location problem

exactly one of the following containments must hold:

$$x_j - x_i \in [l, u] \tag{3.1}$$

$$x_i - x_j \in [l, u] \tag{3.2}$$

These containment choices are represented in the *edge orientation graph*, $G = (V, E)$. The vertices of the graph are the variables x_i. Each separation constraint (i, j, l, u) contributes a directed edge in the graph between x_i and x_j. If (3.1) holds, then the edge is directed from x_i to x_j and is said to be *left oriented.* If (3.2) holds, then the edge is directed from x_j to x_i and is said to be *right oriented.* We allow a pair of vertices to have multiple edges connecting them; this would occur if the same pair of probes occured in several separation contraints.

If the orientation of each edge is specified, then finding the feasible set of probe positions reduces to solving a linear program of a particular form. Solutions to the linear program will also be solutions to the probe location problem instance. If a solution exists, the edge orientation is said to be *feasible.* Thus, the problem is to

find all the feasible edge orientations and their probe position solution sets. The probe positions solution sets can be visualized by a two-dimensional apparatus consisting of vertical rods for the probes and sliding boxes rigidly attached to horizontal rods for the distance constraints (see Figure 3.2). If a probe bar has an attached bead that falls within a box, then the bead is constrained to stay within this box. This fixes the relative orientation of the two probes and constrains the separation distance to fall in the interval determined by the length of the horizontal rod and the size of the box.

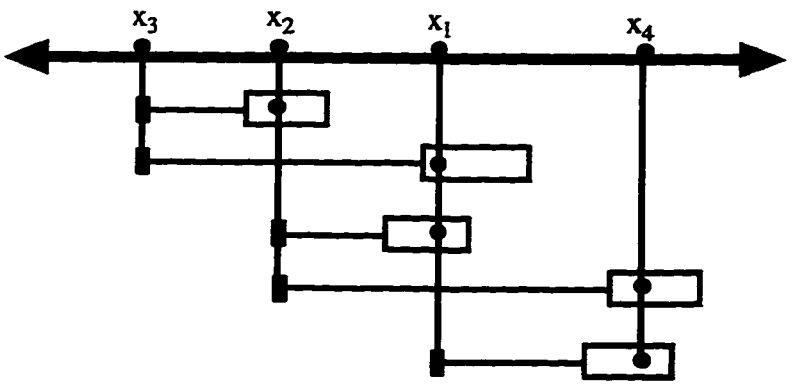


Figure 3.2: Representing probe position solutions

We next show that a standard algorithm can be used to quickly check whether a particular edge orientation is feasible. Each directed edge in the edge orientation graph contributes two such constraints, since the containment $x_j - x_i \in [l, u]$ can be represented as

$$x_j - x_i \leq u \text{ and } x_i - x_j \leq -l.$$

A linear program with constraints entirely of this form has a useful property: It has a feasible solution if and only if there are no negative weight cycles in the edge orientation graph [8]. The weight of a cycle is computed as follows: If an edge with associated distance interval $[l, u]$ is traversed in the forward direction, u is added to

the weight; if the edge is traversed in the backward direction, l is subtracted from the weight. The Bellman-Ford algorithm [8] can be used to check for the existence of negative cycles in $O(nm)$ time, where n is the number of vertices and m is the number of edges. If no negative cycles exist, the Bellman-Ford algorithm outputs a solution vector $(x_1, \ldots, x_n)$ that minimizes $\sum x_i$. This implies the linear program is feasible. The solution space of the linear program represents all probe location solutions which are consistent with the specified edge orientation. The space of all feasible probe locations is thus the union of the solution spaces consistent with all feasible edge orientations.

Actually finding a feasible edge orientation can sometimes be hard, as the following theorem indicates.

Theorem 3.2.1 *PROBE-LOCATION is NP-complete for edge orientation graphs consisting of one large cycle.*

Proof. The proof is via a reduction from the NP-complete problem SET-PARTITION [11]. This is the problem of deciding whether there is a partition of a set of real numbers such that the sums of the numbers in each side of the partition are equal. Let the set of numbers of a particular instance I_S of SET-PARTITION be $\{d_1, \ldots, d_n\}$. Consider the instance I_P of PROBE-LOCATION with probes $\{p_1, \ldots, p_n\}$ and separation constraints $(i, (i \bmod n) + 1, d_i, d_i)$ for $i = 1, \ldots, n$. We will show there is a one-to-one correspondence between solutions of I_S and I_P. Let $(x_1, \ldots, x_n)$ be a solution I_P. We identify it with a unique partition of the $\{d_i\}$ as follows: In the associated edge orientation graph, if e_i is oriented from i to $(i \bmod n) + 1$ then we put i on the left side L of the partition, otherwise if e_i is oriented from $(i \bmod n) + 1$ to i, we put i on the right side R of the partition. Notice that the sum around the cycle is:

$$\begin{aligned} 0 &= (x_1 - x_2) + \ldots + (x_{n-1} - x_n) + (x_n - x_1) \\ &= \sum_{i \in R} (x_i - x_{(i \bmod n)+1}) - \sum_{i \in L} (x_i - x_{(i \bmod n)+1}) \end{aligned}$$

$$= \sum_{i \in R} d_i - \sum_{i \in L} d_i.$$

So (L, R) is a solution of I_S. Likewise any partition (L, R) which is a solution of I_S is identified with a unique edge orientation which is a solution of I_P. The reduction from I_S to I_P can clearly be done in polynomial time, so we conclude that PROBE-LOCATION is NP-complete. Δ

Empirically, the problem becomes easier when additional distance intervals are known. Additional intervals add new edges in the edge orientation graph. New edges decompose large cycles into many smaller cycles, each of which is constrained to be non-negative. These additional constraints help to guide the search for feasible edge orientations. The next section presents an algorithm which relies on this fact to exhaustively find all the feasible edge orientations.

3.3 Finding Feasible Edge Orientations

In the previous section it was shown that PROBE-LOCATION was equivalent to finding the set of feasible edge orientations in the edge orientation graph. We present an algorithm to determine this set. The general idea is to represent edge orientations as paths in a binary tree called the *edge orientation tree*. All the nodes at a particular level in the tree are associated with one edge in $e \in E$. Left branches from nodes at this level fix e to have a left orientation. Right branches fix e to have a right orientation. Each edge in E is assigned a unique level. A depth-first exhaustive search is conducted from the root to find all feasible solution paths. At each new node the Bellman-Ford algorithm is run to determine whether the edge orientations fixed so far are feasible. If not, the subtree rooted at this node is not considered in the search.

The performance of this approach is highly dependent on the order in which edges occur in the edge orientation tree. If a poor edge ordering is used, then many dead-end branches will be explored before a feasible solution path is found. On the other hand, if a good ordering is used, then the search should never stray too far from these

paths. Lemma 3.3.1 shows that if a dead-end branch is taken when the opposite edge orientation leads to a feasible solution, then any infeasible set of edges must contain the edge at the branchpoint.

Lemma 3.3.1 *Let v be the node encountered when the algorithm first strays from a feasible solution path and let e_v be the edge associated with this node. If the edge orientations above v are held fixed, then any set of edges S that cannot be feasibly oriented must include e_v.*

Proof. Let S be a set of edges that cannot be feasibly oriented when the edge orientations above are held fixed (at least one such S must exist as v is not on a solution path). Let p be a solution path that goes through v's parent node. Consider the edge orientation path p' induced by assigning the remaining edges below v the same orientation that p does. Since p provides a feasible orientation of S, and p' and p only differ in their orientation of e_v, it follows that S must contain e_v. Δ

A minimal infeasible set of edges is either a single infeasible cycle or the union of overlapping cycles which are simultaneously infeasible such has shown in Figure 3.3. This suggests that the edges should be ordered in such a way that cycles appear in the edge list as nearly consecutive runs. This way, when the search proceeds down a dead-end path it will be likely to encounter the remaining edges of an infeasible set quickly and be stuck. The following heuristic attempts to order the edges so that this is the case.

3.3.1 The Edge Ordering Heuristic

Assume that the graph is connected. If this is not the case, then each connected component is independent and can be treated separately. The general idea is to order the edges of the subgraph induced by a set of vertices V and incrementally increase the size of V. Initially, V consists of the singleton vertex of greatest degree. A breadth-first search is conducted to find a shortest path which leaves and re-enters

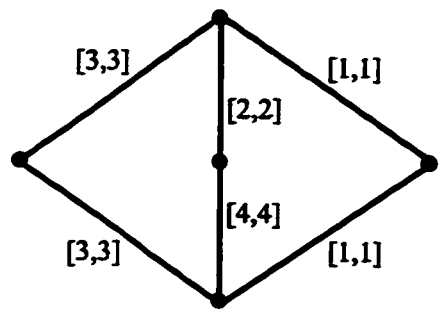


Figure 3.3: Each individual 4-cycles is feasibly orientable, but they cannot be simultaneously oriented. The left cycle requires that the middle edges run in the same direction, while the right cycle requires the middle edge run in opposite directions.

V. If there are any new vertices on this path, they are added to V. The path edges are concatenated to the end of an edge list L. New paths are added in this way until any path of unused edges leaving V does not return. The remaining unused edges are added to the end of L and this list is returned as the edge order.

3.4 Detecting Errors

As mentioned in the introduction, it is possible that some of the distance intervals in the input are bad. With these bad values present, it may be impossible to find a consistent solution. This problem is ubiquitous to combinatorial algorithms that must deal with inconsistencies in real data. We outline a technique for finding a set of bad edges given an infeasible input data set.

We consider the following abstract *suspect-identification* problem. The input consists of a set E of *elements*. Among these elements are some small number of *bad elements*. Associated with the bad elements are subsets of the elements called *gangs*. A gang must contain a bad element and cannot be a proper subset of another gang. The basic computational primitive available is a *gang check* that determines whether a gang is present in a subset of the elements. A bad element must belong to at least

one gang, and may belong to multiple gangs. An element is a *suspect* if it belongs to every gang containing a particular bad element. Elements that are not suspects are called *bystanders*. We initially describe a simple, but impractical deterministic procedure for identifying the suspects in the input that works provided a certain criterion holds. This method suggests a randomized procedure that works in practice, although so far precise bounds on its performance have not been determined. The suspect-identification problem can be used to find and remove bad edges in probe location input data. The edges are the elements; the bad edges are the bad elements. The gangs are all the minimal unorientable subsets of edges (typically unorientable cycles). Checking if a subgraph has a feasible orientation is relatively fast in practice, so it is reasonable to take this as a basic primitive.

We begin with a simple but impractical deterministic suspect-identification procedure. We need the following definition: A *mob* is a union of gangs such that each bad element appears in at least one gang in the mob. The following lemma shows that the set of suspects is the intersection of all the mobs.

Lemma 3.4.1 *An element x is a suspect if and only if it is present in every mob.*

Proof. If x is a suspect, then it is present is every gang containing some bad element b. By definition, any mob must contain at least one of the gangs associated with b, and so must contain x. On the other hand, if x is not a suspect, then for each bad element b there is some gang G_b that does not contain x. Thus, the mob consisting of the union of these G_b will not contain x. Δ

For the procedure to work, the following mob recognition criteria must hold: *If M is a collection of gangs such that $E - M$ is gang-free, then M is a mob.* All the gangs can be found by enumerating all possible subsets S of the elements and checking if each is a gang; this requires checking that S contains a gang and removing any element in S negates this property. All possible mobs can then be found by considering all possible unions of the gangs and checking if the complementary set is gang-free. Since

a mob is a subset of the elements, there are also an exponential number of mobs. If one keeps track of which subsets have been previously examined, all of the mobs. can be enumerated in exponential time (and space). As the mobs are enumerated, elements not occuring in every mob are eliminated as possible suspects. The elements that remain are the suspects.

A randomized procedure for suspect-identification works well in practice. The idea is to repeatedly construct random mobs. The intersection of a certain number of random mobs is reported as a set of *potential suspects*. This set will contain all of the suspects, but may also contain some bystanders. Constructing a random mob M requires a subroutine to find a random gang. The subroutine will be described in the next paragraph. Random gangs are sampled with replacement from E and added to M until $E - M$ is gang-free. Sampling with replacement is needed to insure that every mob has some chance of being sampled.

We need a subroutine to find a single random gang in a set of elements S, provided one exists. First, a random k-way partition of the elements is found, starting with $k = 2$. We consider the subsets $S - S_i$, where S_i is the set of elements in the i-th partition class. We apply the gang check primitive to each of the $S - S_i$ successively. If for some i, $S - S_i$ contains a gang, we set $S = S - S_i$ and recurse. If none of the $S - S_i$ contain a gang, k is incremented by one. If a gang of size s exists, it is guaranteed to be detected when $k = s + 1$. When $k = |S| + 1$, S itself is a gang and is reported. Because a fraction $1/k$ elements are removed each time, a gang of size s will be found with $O(s \log_s |S|)$ gang checks. This compares favorably to the naive approach of checking all size s subsets exhaustively to find a gang. For each gang, there is a positive probability that it is sampled, although smaller gangs are favored by this approach.

In practice, sampling is done without replacement to speed up mob-finding. It seems beneficial that the gang sampling subroutine favors small gangs. This should reduce the size of the potential suspect set more quickly than if large gangs are

picked. Of course, there is a possibility that every small gang includes a non-suspect that is only omitted from a larger gangs. A further complication is that the mob recognition criterion is not always met. A simple example where it is not met is shown in Figure 3.4: a single gang meets the mob-recognition criterion but does not constitute a mob. The mob-recognition criterion may be nearly met, in following sense: For some set of bad elements B, if $E - M$ is gang-free, then for each $b \in B$, M includes a gang containing b. In this case, all of the suspects associated with bad elements in B will be identified as (potential) suspects. Unfortunately, bad elements not in B may be omitted from the (potential) suspect set. We currently address this issue by keeping a count of the number of times each element appears in a bad mob and sorting the elements based on this statistic. In practice, most bystanders will have lower counts than most suspects. Binary search is used to find the shortest prefix of the sorted list of elements whose removal makes the input gang-free. The suspect-recognition problem is an interesting theoretical problem that merits further study.

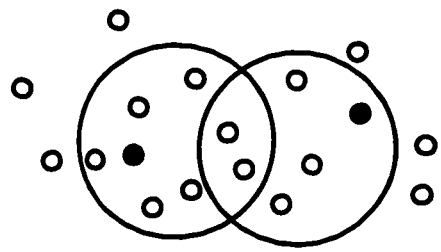


Figure 3.4: A suspect-identification instance for which the mob-recognition criteria does not hold. The black points represent bad elements and the gangs present are circled.

3.5 Experimental Results

In this section, we give the results of some initial experiments with an implementation of the proposed probe-location algorithm. In the program, the input is first tested for feasibility. If it is found to be infeasible, the error-detection algorithm is invoked and the minimum number of bad distance intervals are removed from the input. The set of feasible solutions is reported. Testing was done with synthetic data and FISH distance data from human chromosome 4.

3.5.1 Synthetic Data

One interesting question is which graph structure gives the most map information for a given number of edges? In other words, if one is designing an experiment with no *a priori* knowledge of the probe locations, which pairs of probes should one select to measure and how many are enough? In the following experiment we consider four different types of graph structures built with the same number of vertices and edges:

- *random*: The endpoints of each edge are selected uniformly at random (without replacement).

- *ring*: The vertices are placed in a ring, with edges between consecutive vertices. Any remaining edges are added randomly.

- *1-star*: One vertex is selected at random to be a center. Every other vertex is connected to the center via an edge. Any remaining edges are added randomly.

- *2-star*: Two vertices are selected at random to be centers. Each center is connected to every other vertex. Any remaining edges are added randomly.

To determine how well each graph type performed, a number of random instances of each type were generated for input to the program. In addition to varying the graph

type, the number of edges was also varied to get an idea about the number of probe pairs needed to constrain the solution. Problem instances were generated as follows: First, 40 points were sampled uniformly at random from the real interval $[0, 100]$. Then edges were randomly sampled using the appropriate graph type model. For each edge distance d, the distance interval $[d/(1+s), d(1+s)]$ was used, where s is a fixed *interval tolerance*. In all of the trials, s was set to 0.1. Hence an actual distance of 10 is supplied to the program as the distance interval $[9.09, 11.0]$. Table 3.1 presents the results. Each input combination was run twenty times and collective statistics were tabulated. The *ambiguous probe pairs* column gives the average of the logarithm (base 2) of the number of different feasible edge orientations. The reason for taking the logarithm is that the number of feasible edge orientations should be roughly two to the number of ambiguous probe pair orientations, provided they are mostly independent. The *tree nodes* column gives the geometric mean of the number of nodes examined while searching the edge orientation tree and is roughly proportional to the running time. The number of nodes examined should be roughly proportional to the number of feasible edge orientation paths. The distribution of the number of feasible paths should be roughly the exponential of the distribution of the number of probe pair amibiguities, and so the geometric mean seems appropriate. The *mean displacement* gives the mean of positional displacement error between the computed probe positions (the solution vector $(x_1, \ldots, x_n)$ produced by the Bellman-Ford algorithm) and the true probe positions. Position comparison was done by aligning the first probe in the computed map with the first probe in the true map and choosing the orientation of the computed map to minimize the sum of the displacement errors. The mean is taken over all runs for each input combination.

To test the performance of the error detection method, 10 edges were chosen at random to be bad edges. For 5 of these edges, if the true distance of the edge was d then the bad distance interval was constructed as if the distance was $d/2$. For the other 5, if the true distance of the edge was d then the distance interval was

Table 3.1: Experimental results for error-free data sets (40 probes)

number of edges	graph type	tree nodes	ambiguous probe pairs	mean displacement
100	random	3458	6.1	22.3
100	ring	6349	6.0	8.4
100	1-star	1698	5.5	17.0
100	2-star	1362	4.0	6.9
200	random	3972	3.8	3.6
200	ring	3981	3.8	3.7
200	1-star	3309	4.0	3.2
200	2-star	2722	3.5	3.7

constructed as if the distance was $2d$. Table 3.2 presents the results averaged over twenty independent runs. The number of edges was fixed at 200. The *bad edges accepted* column gives the average number of bad edges that were not excluded from the input. The *good edges removed* column gives the average number of good edges which were removed from the input.

The results do not strongly favor any one type of graph. If there are fewer edges and error-free data it appears that the *2-star* topology is the best. On the other hand, if more edges are provided and the data may contain errors the *1-star* topology appears slightly superior. The mean displacements are improved (lowered) with additional edges. This indicates that the solution space becomes more constrained with additional edges, as one would expect.

Table 3.2: Experimental results for data sets with errors (40 probes, 200 edges)

graph type	tree nodes	ambig. probe pairs	bad edges accepted	good edges removed	mean displacement
random	3206	4.0	0.9	0.2	4.0
ring	5211	4.2	1.2	0.6	4.2
1-star	2802	3.2	0.5	0.2	3.2
2-star	2374	3.6	1.2	0.5	3.6

3.5.2 Chromosome 4 Data

The next data set considered was based on FISH distance measurements for 10 cosmid probes on band $p16$ of chromosome 4.[2] Maps of these probes in this 4 Mb region have been published in the literature (see [28] for a list of references). The published reference maps agree on the probe order, but the inter-probe distances reported vary between the maps by 10 to 30%. The data was collected as follows: Interphase distance measurements were made for forty-four pairs of probes. Each pair considered was measured approximately 100 times. For each pair, the mean of the squared measurements was computed (the random-walk model predicts the genomic distance to be linear in this quantity). The mean values were converted to distance intervals by assuming the true distance to lie within 30% of the measured distance in either direction. Previous studies [28, 27, 30] have shown this corresponds to a high confidence interval (at least 90%) for genomic distances up to about 1.5 Mb. These distance intervals were supplied as input to the program. Only one solution was found, and it agreed with the established probe order and approximate interprobe distances. It

[2] Data set supplied by B. Trask, Dept. of Molecular Biotechnology, Univ. of Washington.

was interesting that it was necessary to discard about 20% of the intervals in order to find a consistent solution. If the interval tolerance was lowered to 25% a few more intervals were discarded but the same general solution was found. Likewise, if the tolerance was raised to 35%, the same solution was found, with a few less intervals discarded. The discarded edges were mostly too short to fit into the solution map, which predicted their separation to be larger. One explanation would be that the chromosome is folding back on itself more than a random walk would predict. Thus, for some pairs of probes, the genomic separation distance is underestimated by the random-walk model. We discuss this more in the next section.

3.6 Future Work

Three other problems for which the probe-location algorithm could be useful are mentioned. The issue of generalizing the pairwise distance error model is also discussed.

3.6.1 Mapping DNA Backbone Structure

It has been proposed [25] that nucleic DNA folds along a protein backbone structure which itself behaves like a random walk [30]. Along the backbone, loops of size 1-2 Mb occur periodically (see Figure 3.5). The physical distance separating a pair of probes is described by a random walk of length d, where d is the genomic distance separating the probes, where possible shortcuts along the backbone are taken. Thus for pairs of probes which are both close to the backbone (the black probes in the figure), the distance separating them along the backbone can be estimated from FISH distance measurements. For these probes it should be possible to build a consistent linear map using the probe-location algorithm. It may not be possible to consistently place the probes which are further away from the backbone (white in the figure) into this map. This is because physical distances between white and black probes do not follow a simple random-walk model. These inconsistencies should be detectable by the error-

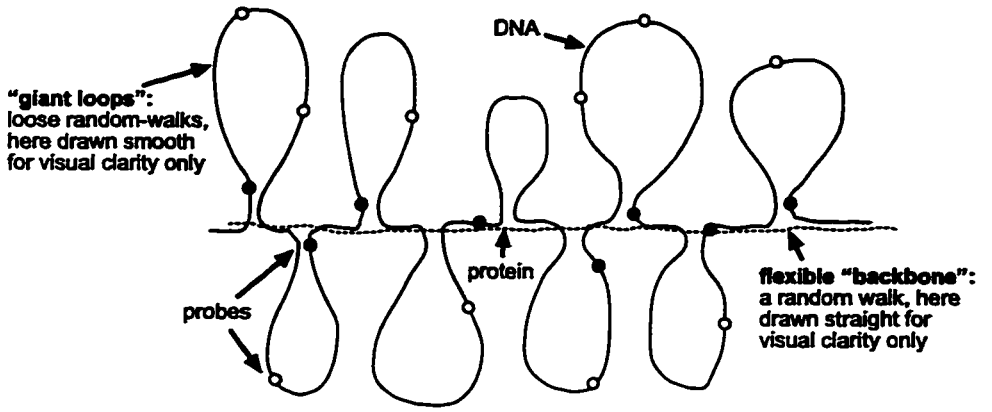


Figure 3.5: DNA folding along a random walk backbone

detection algorithm. One would expect that white probes would participate in more bad edges than black probes. If this were the case, then they could detected and removed. The remaining probes would be black and a map of their positions along the backbone structure could be determinded by the probe-location algorithm.

3.6.2 Consensus Maps

Another application is to the problem of building *consensus maps*. If several maps have been made for the same region of the genome, it would nice to know if they are mutually consistent, and if so what total constraints on the positions of the markers are imposed. Each map provides distance constraints for some set of location markers. Typically these distance constraints can be written in the form of confidence intervals. For example, distances can be estimated in restriction fragment maps by summing the lengths of restriction fragments known to fall between a pair of markers. For STS content maps made with uniformly sized clones, distances can be estimated by the number of clones separating two markers. The constraints that each map imposes could be considered together to find a consensus map. If no solution exists, the error-detection algorithm could be used to locate the inconsistencies for further checking.

3.6.3 Ordering Clone Libraries

The probe-location algorithm could be useful in assisting the mapping of clone libraries, provided the mapmaking process provides distance information between pairs of clones. In constructing restriction fragment maps, contigs of clones that overlap substantially are first found. Distances between pairs of clones in each contig can be estimated by summing up the lengths of fragments postulated to lie between the clones. If multiple restriction enzymes are used, the contigs found in each enzyme digestion will, in general, be different. The probe-location algorithm could find clone orderings simultaneously compatible with pairwise distance data estimated from each enzyme digestion.

3.6.4 Generalizing the Distance Error Model

It would also be interesting to consider generalizations to the distance error model. A *biased confidence interval model* would possibly be relevant to mapping a genomic region that behaved mostly like a random walk, but occasionally looped back upon itself. In this model, the probability that the confidence interval underestimates the true genomic distance is greater than the probability that it overestimates. In other words, it is more likely that the true distance falls beyond an inconsistent distance interval, rather than falls short of the interval. These error priors could be used in deciding which edges to remove in the error-detection phase. Finally, it would be interesting to consider more general models of pairwise measurement error. If the error distribution function was known, one could search for a maximum likelihood probe positioning. This would invariably be more computationally intensive, and might be intractable for realistic problem sizes.

Chapter 4

CONCLUSIONS

In this thesis, I have made research contributions on two computational problems in genomic mapping. The clone ordering and overlap algorithms relied on a novel calculation of posterior overlap probabilities and local search heuristics to determine good clone orderings. The probe location problem could be cast as an graph edge orientation problem and a randomized technique was developed to detect and remove inconsistencies in the input. Each of the algorithms was implemented and performed well on both synthetic and real data sets.

BIBLIOGRAPHY

[1] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. Watson. *Molecular Biology of the Cell.* Garland, 1994.

[2] Alizadeh, Karp, Weisser, and Zweig. Physical mapping of chromosomes using unique probes. In *ACM-SIAM Symposium on Discrete Algorithms*, 1994.

[3] F. Alizadeh, R. M. Karp, L. A. Newberg, and D. K. Weisser. Physical mapping of chromosomes: A combinatorial problem in molecular biology. *ACM-SIAM Symposium on Discrete Algorithms*, pages 371–381, 1993.

[4] F. Alizadeh, R. M. Karp, L. A. Newberg, and D. K. Weisser. Physical mapping of chromosomes: a combinatorial problem in molecular biology. *Algorithmica*, 13(1/2):52–76, January 1995.

[5] R. Arratia, E. Lander, S. Tavare, and M. Waterman. Genomic mapping by anchoring random clones: A mathematical analysis. *Genomics*, 11, 1991.

[6] D. Balding and D. Torney. Statistical analysis of DNA fingerprint data for ordered clone physical mapping of human chromosomes. *Bulletin of Mathematical Biology*, 53(6):620–629, 1991.

[7] K. H. Buetow and A. Chakravarti. Multipoint gene mapping using seriation. i. general methods. *Am. J. Hum. Genetics*, 41:180–188, 1987.

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, 1990.

[9] D. P. Fasulo, T. Jiang, R. M. Karp, R. Settergren, and E. C. Thayer. An algorithmic approach to multiple complete digest mapping. In *Proceeding of the First Annual International Conference on Computational Molecular Biology*, 1997.

[10] J. Freund and R. Warpole. *Mathematical Statistics*. Prentice Hall, 1987.

[11] M. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. Freeman and Co., San Francisco, 1979.

[12] W. Gillet and J. Daues. DNA mapping algorithms: Synchronized double digest mapping. Technical report, Washington University, 1993.

[13] D. Greenberg and S. Istrail. The chimeric mapping problem: Algorithmic strategies and performance evaluation on synthetic genomic data. *J. Computers and Chemistry*, 1994.

[14] T. Hudson, L. Stein, S. Gerety, J. Ma, A. Castle, J. Silva, D. Slonim, R. Baptista, L. Kruglyak, S. Xu, X. Hu, A. Colbert, C. Rosenberg, MP. Reeve-Daly, S. Rozen, L. Hui, X. Wu, C. Vestergaard, K. Wilson, J. Bae, S. Maitra, S. Ganiatsas, C. Evans, M. DeAngelis, K. Ingalls, R. Nahf, L Horton, M. Oskin, A. Collymore, W. Ye, V. Kouyoumjian, I. Zernsteva, J. Tarn, R. Devine, D. Courtney, M. Renaud, H. Nguyen, T. O'Connor, C. Fizames, S. Faure, G. Gyapay, C. Dib, J. Morissette, J. Orlin, B. Birren, N. Goodman, J. Weissenbach, T. Hawkins, S. Foote, D. Page, , and E. Lander. An STS-based map of the human genome. *Science*, 270:1945–1954, 1995.

[15] S. Istrail. The chimeric clones problem. Technical report, Sandia National Laboratories, 1993.

[16] T. Jiang and R. M. Karp. Mapping clones with a given ordering or interleaving. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1997.

[17] N. L. Johnson and S. Kotz. *Urn Models and Their Application.* Wiley, 1977.

[18] Kamath, Motwani, Palem, and Spirakis. Tail bounds for occupancy and the satisfiability threshold conjecture. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1994.

[19] Alan Karr. *Probability.* Springer Verlag, 1992.

[20] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[21] B. Mumey. A fast heuristic algorithm for a probe location problem. In *Proceedings of the Fifth Annual Conference on Intelligent System for Molecular Biology*, 1997.

[22] D. O. Nelson and T. P. Speed. Statistical issues in constructing high resolution physical maps. *Statistical Science*, 9(3):334–354, 1994.

[23] B. Pinkerton, 1993. Results of a simulated annealing algorithm for FISH mapping. Communicated by Dr. Larry Ruzzo, University of Washington.

[24] J. Redstone. Algorithms for ordering DNA probes on chromosomes, 1996. Ph.D. Qualifying Exam paper. Department of Computer Science, University of Washington.

[25] R. K. Sachs, G. van den Engh, B. Trask, H. Yokota, and J. E. Hearst. A random-walk / giant-loop model for interphase chromosomes. *Proc. Natl. Acad. Sci. USA. Biochemistry*, 92:2710–2714, 1995.

[26] C. Soderlund, D. Torney, and C. Burks. Calculating shared fragments for the single digest problem. In *Proceeding of the Twenty-Sixth Hawaii International Conference on System Sciences. Wailea, HI, USA.*, volume 1, 1993.

[27] B. J. Trask, S. Allen, H. Massa, A. Fertitta, R. Sachs, G. van den Engh, and M. Wu. Studies of metaphase and interphase chromosomes using fluorescence in situ hybridization. *Cold Spring Harbor Symposia on Quantitative Biology*, LVIII:767–775, 1993.

[28] G. van den Engh, R. K. Sachs, and B. J. Trask. Estimating genomic distance by a random walk model. *Science*, 257:1410–1412, 1992.

[29] J. Watson, M. Gilman, J. Witkowksi, and M. Zoller. *Recombinant DNA*. Scientific American Books, 1992.

[30] H. Yokota, G. van den Engh, J. E. Hearst, R. K. Sachs, and B. J. Trask. Evidence for the organization of chromatin in megabase pair-size loops arranged along a random walk path in the human g0/g1 interphase nucleus. *The Journal of Cell Biology*, 130(6):1239–1249, 1995.

Brendan Marshall Mumey

Date of birth: 12 July 1968
Countries of citizenship: Canada and U.S.

Department of Computer Science and Engineering
University of Washington
Box 352350
Seattle, WA, 98195-2350

Education

Ph.D., Computer Science (May 1997)
University of Washington
Thesis title: "Some Computational Problems from Genomic Mapping."
Committee: Dr. Larry Ruzzo, Dr. Richard Karp, and Dr. Richard Anderson

M.Sc., Computer Science (August 1992)
University of British Columbia
Thesis title: "Some New Results on Constructing Optimal Alphabetic Binary Trees."
Advisor: Dr. Maria Klawe

B.Sc. (Honors First Class), Mathematics (May 1990)
University of Alberta

Publications

1. B. Mumey, "A Fast Heuristic Algorithm for a Probe Mapping Problem." Presented at *Intelligent Systems for Molecular Biology 1997.*

2. B. Mumey, "Finding Clone Overlaps and Orderings from Restriction Fragment Fingerprints." Presented at *Intelligent Systems for Molecular Biology 1996.* Journal version in preparation.

3. B. Mumey, "Finding Friends-Of-Friends Clusters Quickly." Appears in *Parallel Algorithms: Third Annual DIMACS Implementation Challenge.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Volume 30. American Mathematical Society.

4. M. Klawe and B. Mumey, "Upper and Lower Bounds on Constructing Optimal Alphabetic Trees." *Proceedings of the 1993 ACM/SIAM Symposium on Discrete Algorithms. January 1993. SIAM Journal on Discrete Math. November 1995.*