Construction of High-Resolution Likelihood-Based Integrated Genetic and Physical Map of *NEUROSPORA CRASSA*

by

Susanta Tewari

(Under the direction of Jonathan Arnold and Suchendra M. Bhandarkar)

Abstract

Neurospora crassa, a model organism has been extensively used to explore fundamental cellular processes, such as recombination and for construction of high-resolution maps because of its excellent biological characteristics. In this thesis we bring to bear statistical techniques to integrate genome map information gleaned from two different sources with varying resolution to create a high resolution integrated physical and genetic map. Those two different sources of genomic information in our case are a high resolution restriction frequent length polymorphism (RFLP) DNA markers and hybridization based physical data. Recombination has long been used to create genetic maps starting from the rudimentary but powerful empirical pair-wise frequency analysis to sophisticated statistical models with generalized cross-over phenomena. Most of these approaches are theoretical lacking any practical model estimation, or fail to capture the details of the crossover process. In recent times the need has come up to analyze more number of genes to answer more complex problems as gene discovery and disease tracing. In the first chapter we give a synopsis of the entire work. In the second chapter of the thesis we have included published work that formulates a detailed mathematical model of the recombination process. In the third chapter we detail on a novel recursive linking algorithm that overcomes a computational bottleneck often faced in gene mapping, the exponential time complexity of the algorithm in the number of markers. In the fourth chapter we integrate physical and genetic maps to produce a high resolution integrated physical and genetic map and study various genomic questions. For example, we construct empirical mapping functions that relate the amount of genetic recombination to physical distance. At the end we study the distribution of repeated DNA markers and outline the potential for progress in future endeavors.

INDEX WORDS: Genetic Map,Physical Map,RFLP Data,Hybridization Data,Recombination, Simulated Annealing, Maximum Likelihood Estimator, Whole Genome-shotgun Sequencing,Tetrad Analysis,EM Algorithm,Recursive Linking,Time Complexity

Construction of High-Resolution Likelihood-Based Integrated Genetic and Physical Map of *NEUROSPORA CRASSA*

by

Susanta Tewari

B.S., Haldia Govt. College, Vidyasagar University, 2000

M.S., Department of Statistics , University of Pune, 2002

A Thesis Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2008

© 2008

Susanta Tewari

All Rights Reserved

Construction of High-Resolution Likelihood-Based Integrated Genetic and Physical Map of *NEUROSPORA CRASSA*

by

Susanta Tewari

Approved:

Major Professors:	Jonathan Arnold
	Suchendra M. Bhandarkar
Committee:	Gauri S. Datta
	Nicole Lazar
	Paul Schliekelman
	T. N. Sriram

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2008

Dedication

To *Baba* and *Ma* who are my first inspiration,

Acknowledgments

I would not be writing this thesis if it was not for an email advertisement sent out by one of my co-advisors Dr. Suchedra M. Bhandarkar asking for a student in Statistics who would be interested in a cross disciplinary work. Then I hardly knew the term genetic mapping. Nevertheless, I felt that this could be an opportunity to enter into a deterministic field and see how a non-deterministic field such as statistics may be used successfully. The motivation to affect as applied a field as biology through statistics was overwhelming and I remember I turned to one of my committee members Dr. Gauri Datta to seek advice on this. His encouragement reflected my intuition and before long I was into this exciting field *Bioinformatics*. Today, if I say that Bioinformatics is exciting, it is largely because of the excitement with which my advisor Dr. Jonathan Arnold guided me throughout my research. He gave me plenty of time to think and provided pointers that helped enormously. He created an environment where research became more like an adventure and I could feel the team spirit as we shared the excitement and disappointment side by side. I thank Dr. Suchedra Bhandarkar for being very patient with me and allowing me to enjoy the work. He kept me on toes, providing the practical tips and helping me avoid the common problems a budding researcher faces.

During the early years in the statistics department at UGA when I was preparing the background for a PhD, I learned from many and I fondly remember Dr. T.N. Sriram, Dr. Nicole Lazar, Dr. Gauri Datta, Dr. Daniel Hall, Dr. Jaxk reeves, Dr. Lynne Syemour and Dr. Yin among others. I thank graduate school for awarding me Dissertation Award which allowed more uninterrupted time for research. I am thankful to Dr. John Stufken for helping me fund JSM travels. The research was supported in part by a grant from the U.S. Dept. of Agriculture under the NRI Competitive Grants Program (Award No: GEO-2002-03590) to Drs. Bhandarkar and Arnold. I am grateful to Dr. Paul Schliekelman, Dr. Gauri Datta, Dr. T.N. Sriram and Dr. Nicole Lazar for serving as committee members.

I would like to take this unique opportunity to thank people that helped me become the way I am. My father and mother were the first sources of inspiration for me and remain a guiding force in whatever I do. Before I knew what this word *inspiration* meant I felt it as I saw my father work too hard to meet the loose ends for the family. My mother with her little formal education managed the household like magic and provided endless sacrifices which helped me later understand the word *love*. My sister Sampa and brother Gopal made growing up a wonderful experience. During the start of my college years in my hometown *Haldia*, I met my friend Subho who opened the world for me. I learned many things from him and spent an unforgettable time. My other two loving friends Pratap and Sudip made the college years relaxing and I fondly remember how we would all gather up at Sudip‘s place without telling him beforehand and either cook and have lunch at his place or just go out. I dearly remember the time spent as a private tutor of *Moutushi* at Paik‘s in our colony. I specially remember the good food that I would be insisted upon every time I visited. I miss my first income Rs:300 from the tuition which I had great difficulty spending as I kept on permuting the order of items in my wish-list. Soon I became friends with the whole family and little did I know that I would become a part of their family.

During my school years I remember Bhupal Sir and my private tutor Pahari *Mastermoshai* for instilling a passion for mathematics. In college I learnt most from Prof. Sib Narayan Guria and Prof. Tapan Kumar Pal. During my M.Sc in the Department of Statistics, Pune University I came close to many inspiring and talented teachers. Among them are Prof. U.V. Naik-Nimbalkar, Prof. M.B. Rajarshi, Prof. S.R.Deshmukh., Prof.J. V. Deshpande., Prof. S. R. Paranjape, Prof. S. Kunte., Prof. A. V. Kharshikar and Prof. A. P. Gore. I found many friends from different cultural and socio-economic backgrounds and they all made the experience at Pune enriching. In athens, I most fondly remember Banerjee-da, Chowdury-da and Bhatchaz-da for their simplicity of heart and uncomplicated manners. Special mention goes to the late night *addas* at our *Parar-Morer-Benchi* where the trio chatted while the others wondered. Baishalli-di indulged me on delicious meals and *mashima* taught me some survivor‘s recipes.

Last but closest to heart, is that girl at Paik‘s, my loving wife *Mou*. At the days end, my heart goes back to a time in the past and tries to recapture that look in her eyes, that said, *I Love You*.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Literature Review

Genetic mapping has been attempted ever since an understanding of the recombination took place as early as 1913 [60]. These maps play central roles in understanding the whole biological system in much the same way the map of earth has played a critical role in its exploration. From the production of proteins in cells, their interaction to produce complex molecules such as ribosomes, spliceosomes and DNA polymerase to the ways these complexes interact with one another in sequential reactions to produce pathways and processes which in turn as a whole form the phenotype of the individual genes play a central role, and an accurate map is the only way to ever hope for an full understanding of the biological systems. Recent advances in technology has enabled researchers to move from phenotypic or functional interest in a small set of genes to more widely available feature rich set of markers, known as DNA markers. The advantages of this shift from local to global approach of analysis is manifold. First, obtaining a set of ordered clones of an entire chromosome or an entire genome is an invaluable baseline for future molecular studies of any type. For example, these clones can be used for finding and manipulating the individual genes of interest. Second, genomic DNA is the blueprint of a species, the information needed to build a living cell and a living organism. Hence knowledge of its specific nucleotide sequence lays the groundwork for eventually providing answers to one of the basic questions of biology : *why is an organism the way it is and what makes it different from other organisms ?*.

Chromosomal maps fall into two broad categories- *genetic maps* and *physical maps*. Genetic maps represent an ordering of genetic markers along a chromosome where the distance between two genetic markers is related to their recombination frequency. Genetic maps are typically of low resolution *i.e.*, 1-10Mb [39] in humans. While genetic maps enable a scientist to narrow the search for genes to a particular chromosomal region, it is a physical map that ultimately allows the recovery and molecular manipulation of genes of interest. A physical map is defined as an ordering of distinguishable (*i.e.*, sequenced) DNA fragments called *clones* by their position along the entire chromosome where the clones may or may not contain the genetic markers.

Early efforts in genetic mapping either fail to model in detail the recombination process [39] or provide a such a model without the technology needed to implement it for a large number of markers ([73], [75], [22]). A detailed analysis of the competing models is given in Section 2.4. In this thesis we provide such a model and fill the gap by providing the technology to fit it to data on hundreds of markers. This has long been a challenge because of the inherent exponential nature of the recombination configurations which in turn would appear to give rise to an algorithm of exponential time complexity. We have proposed a novel *recursive linking* algorithm that is inspired by the divide and conquer strategy of a dynamic programming algorithm which reduces the time complexity from exponential to linear. This has enabled us to integrate successfully genetic map with large number of markers with the available likelihood based physical map [8]. This integrated physical and gentic map now provides a single place with a *master-detail* view of both genes and the clones they are tagged to. This map or the maps created this way will be invaluable to the scientific community at large as such maps have an associated probability metric that can be used to measure the relative quality of the map and thus paves the way for better map creation.

1.2 Organization of this Thesis

In chapter 2 we develop a probabilistic framework for the chromatid exchanges for 2 markers and illustrate various characteristics of the model. This model is also used to deal with the tetrad types and we analyze a small data set to compare the traditional estimators of recombination with the ones derived from this model. Also our model agrees with known biological facts (See Section 2.6.3). We develop the likelihood model and use maximum likelihood estimation to estimate the parameter c, the exchange probability between a pair of markers. Detailed proofs of the theorems appear at the end in an appendix. We show the unbiasedness of the estimator of c and address identifiability issues. We study the efficiency of our MLE based estimator with other well known estimators and show its superiority. We extend the model to account for arbitrary number of genes and use the well known EM algorithm to estimate the variable number of exchange parameters simultaneously. We estimate the standard errors of the parameters using rates of convergence of the EM algorithm [44]. A combinatorial optimization technique simulated annealing is used to search of the space of all possible marker orders. We use the high density of RFLP DNA markers in *N.crassa* to create a genetic map and compare it with published maps. Lastly,we provide a goodness of fit measure for the model and show that our model performs modestly well for most of the linkage groups.

As mentioned in Section 1.1, our aim was to provide a detailed model with efficient implementation lacking in the current literature. A straightforward algorithm to solve the EM equations turn out to have exponential time complexity without bound in the number of genetic markers. In this chapter we show in detail with pseudocode the outline of a such a possible algorithm and compute its time complexity. We show with runtime results that fitting a genetic map for more than 7 markers is not feasible.

We propose a novel recursive linking algorithm that is capable of dividing the sequence of marker intervals into variable length intervals and manage the EM calculations for that block and reconstruct the required computation in an exponential manner as one moves along the order of the markers. This efficiently reduces the time complexity from exponential to linear. We give in detail the various stages of this algorithm, illustrating the concepts with diagrams and pseudocode. We calculate the time complexity of the algorithm and show the interval length that gives the best performance. We give runtime results and make a side by side comparison with the straightforward algorithm that immediately shows the usefulness of the algorithm. We also discuss the space and time complexity trade-offs of the algorithm.

Since a straightforward algorithm faced difficulty in solving the EM equations themselves, using it to estimate the standard errors, which is based on the rate of convergence of EM iterations could only result in worse performance. Again, we use the recursive linking algorithm to overcome this problem and discuss the important parts. Simulated annealing is run to create a genetic maps for the whole genome of *Neurospora crassa* which is also used in reporting the map statistics in section 2.15.2. In the end we explore the distribution of the DNA repeats along each chromosome which has important biological significance and discuss how the efforts that have been undertaken has given us new insights for future exploration.

A likelihood based physical map has been available from earlier attempts [8]. Having developed a likelihood based genetic map we were able to integrate the information obtained from two distinct sources and create a high resolution genome map for *N. crassa*. We started with a legacy order of probes on a physical map as an initialization and obtained a marker order based on the sequence based tagging information. We formed a integrated likelihood by combining the log-likelihoods of physical map and the genetic map. Simulated annealing was used to obtain the best integrated orders. An integrated map allowed us to create an empirical mapping function relating the recombination distance to the physical distance. We explore the "centromere" effects on the physical distance per map unit, as well as the distribution of DNA repeats in the *N. crassa* genome with the integrated map.

The concluding chapter ties together the results from the previous chapters and shows the future directions.

Chapter 2

LIKELIHOOD OF A PARTICULAR ORDER OF GENETIC MARKERS AND THE CONSTRUCTION OF GENETIC MAPS [1]

[1] Tewari,S., Arnold, J. and Bhandarkar,S.M. 2008
Journal of Bioinformatics and Computational Biology. 6 : 125-162
Reprinted here with permission of publisher

2.1 Abstract

We model the recombination process of fungal systems via chromatid exchange in meiosis that accounts for any type of bivalent configuration in a genetic interval in any specified order of genetic markers for both random spore and tetrad data. First, a probability model framework has been developed for 2 genes and then generalized for arbitrary number of genes. Maximum Likelihood Estimators (MLE) for both random and tetrad data are developed. It has been shown that the MLE estimator of recombination for tetrad data is uniformly more efficient over that from random spore data by a factor of at least 4 usually. The MLE for the generalized probability framework has been computed using the EM algorithm. Pearson chi-squared statistic is computed as a measure of goodness-of-fit using a product multinomial set-up. We implement our model with genetic marker data on the whole genome of *Neurospora crassa*. Simulated annealing is used to search for the best order of genetic markers for each chromosome, and the goodness-of-fit value is evaluated for model assumptions. Inferred map orders are corroborated by genomic sequence with the exception of linkage group I, II and V.

2.2 Introduction

Since almost the beginning of genetics, an important goal has been to create maps of entire chromosomes. These maps fall into two classes, genetic and physical maps ([16], [30]). The former are constructed from information on how genes are transmitted from parent to offspring. The latter are constructed by having an experimental approach designed to distinguish DNA fragments and to order these fragments. Computationally feasible maximum likelihood solutions for these respective problems were developed by Lander and Green for a genetic map with many markers [39] and by Bhandarkar *et al.* [8] and Kececioglu *et al.* [33] for a physical map composed of many DNA fragments.The focus here is on constructing genetic maps with many markers.

With l markers or genetic loci, each with two or more alternate types of a gene called alleles, the number of possible types of offspring is 2^l and hence the computational complexity of a likelihood-based approach to estimating a genetic map would appear to go up at least as $O(25^l)$ [62]. At first glance the computational complexity of the segregation of l markers to build a genetic map seems daunting. For a special case, Lander and Green [39] were able to develop a maximum likelihood procedure for ordering genetic markers that had a computational complexity linear in the number of markers. This is essential because geneticists have just completed the International HapMap [15] to hunt down most disease causing genes with thousands of markers scattered through the genome and the single nucleotide polymorphisms (SNPs) from the affymetrix chips in complex trait analysis are derived in part from the HapMap SNPs. Several model systems now possess genetic maps with thousands of markers, and their transmission to offspring can be observed simultaneously [70]. In essence simultaneous use of mapping data together with variation in a complex trait, such as many human diseases, provides a triangulation on genes that may influence a complex trait controlled by two or more genes [40].

The ideal data obtainable from a geneticist's perspective, is a situation in which he/she can observe the gametes of a parent directly (as opposed to the offspring) to understand the transmission of genes or genetic markers. Then the transmission of genes in one parent does not mask how genes are transmitted in the other parent. In fungi, such as *Neurospora crassa*, the gametic products can be typed directly (Figure 2.1). These strings of spores are the gametes from a single cross. It is possible to engineer other organisms, such as the model plant system *Arabidopsis thaliana*, for tetrad analysis([18], [23]). The question is for this ideal kind of genetic data (in the best of all possible experimental worlds) can we construct a genetic map with many markers?

Given the importance of tetrads to understanding how genes are transmitted together or separately in the hundreds of fungal laboratories employing these kinds of genetic analyses, you would think there would be a clear statistical methodology for analysis of such multilocus data and the planning of such experiments. The problem examined here is very old, difficult, and solved here for a case relevant to the fungal kingdom and organisms lucky enough to be engineered to have tetrads. We extend the work of Zhao and Speed [75] to the case of many markers [62]. Tetrad data are the best available for constructing genetic maps.

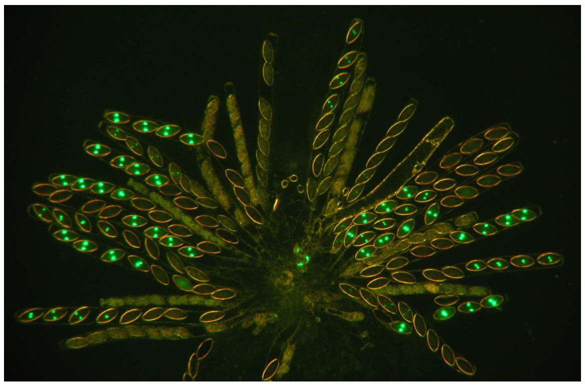


Figure 2.1: Maturing asci of *Neurospora crassa* from wild type x histone $H1-GFP$ (inserted at $his-3$). Histone $H1$, being a chromosomal protein, allows the GFP-tagged nuclei (two per spore at this stage) to fluoresce in four of the eight ascospores; the remaining four ascospores carry the untagged nuclei from the wild-type parent. Almost all asci show the first-division segregation of $hH1-GFP$ because of its close proximity to the centromere of linkage group I. (Photo courtesy of Namboori B. Raju, Stanford University.)

In a previous paper we have demonstrated how the likelihood function involving hundreds of genetic markers can be computed with a computational complexity linear in l [62]. Here we show that the inference tools based on the likelihood function for a genetic map in fact produces the correct map. Resulting genetic maps are independently verified against the sequence of the *Neurospora crassa* genome [24] with the exception of linkage group I, II and V. Inferred recombination distances between markers and their standard errors are reported for the first time for this model system.

2.3 Background Material on Meiosis, Recombination and Crossover

In eukaryotic organisms, the vast majority of genes are found on the chromosomes in the cell's nucleus. Many eukaryotic species are classified as either *diploid*, carrying two nearly identical pairs of nuclear chromosomes (*i.e.*, two nuclear genomes) in each cell, or *haploid*, with only one chromosome set per cell. Most fungi and algae are haploids, whereas many other eukaryotes, including animals and flowering plants, are diploids. However, it is worth noting that diploid organisms produce haploid reproductive cells (such as eggs and sperm in animals); conversely, some haploid organisms, such as fungi, produce specialized diploid cells during the sexual phase of their life cycles. The letter n is used to designate the number of distinct chromosomes in one nuclear genome, so the haploid condition is designated n (that is, $1 \times n$) and the diploid state in $2n$ (that is, $2 \times n$). The symbol n is called the haploid chromosome number.

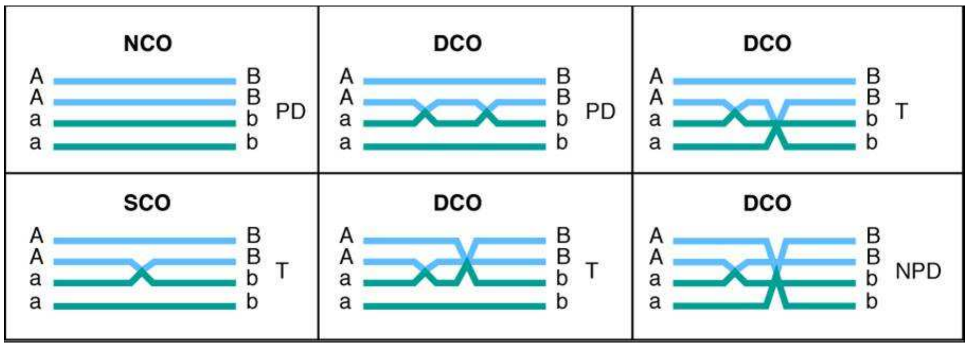


Figure 2.2: The ascus classes produced by crossovers between linked loci. NCO, noncrossover meioses; SCO, single-crossover meioses; DCO, double-crossover meioses. Taken from [63].

In many familiar eukaryotes, recombination principally takes place in *meiosis*. In eukaryotes the sexual cycle requires the production of specialized haploid cells (for example, egg and sperm) called *gametes*. This is achieved by DNA replication prior to meiosis in a diploid meiocyte, followed by two successive cell divisions, resulting in a tetrad of four haploid products or gametes. The two divisions of the nucleus that produce the tetrad of haploid gametes are called *meiosis*. Because of DNA replication prior to meiosis and pairing of homologs, each chromosomal type is represented in four copies, called *chromatids*, or in other words, two pairs of sister chromatids. The two pairs of sister chromatids align, constituting a *bivalent*, or group of four chromatids. It is at this stage that crossing-over is thought to take place. For any particular bivalent, there can be from one to several crossovers. A crossover can be represented by a double-stranded break in which ends of chromatids reanneal with the wrong chromatid ends as shown in Figure 2.2. The crossovers can occur at any position along the chromatids, and the positions are different in different gametes. Furthermore, crossovers are usually only observed between nonsister chromatids, different colors in Figure(2.2). If we designate the sister chromatids from one parent as 1 and 2, and from the other parent, 3 and 4, crossovers can be seen between 1 and 3, 1 and 4, 2 and 3, and 2 and 4 as shown in different crossovers in Figure(2.2). It is plausible that crossovers are equally likely over these pairs, being random events. This assumption is sometimes referred to as *No-Chromatid-Interference*(NCI).

The process underlying meiotic recombination shuffles heterozygous allele pairs and deals them out in different combinations into the products of meiosis (such as the gametes of plants and animals). Being precise, meiotic recombination is defined as the production of haploid products of meiosis with genotypes differing from both the haploid genotypes that originally combined to form the diploid parental meiocyte. The product of meiosis so generated is called a *recombinant*, and the process generating the recombinant is hypothesized to be *crossing-over*. Thus crossing-over is essentially a breakage-and-rejoining process between homologous DNA double helices, in meiosis.

There are two different mechanisms of meiotic recombination: *independent assortment* of heterozygous genes on different chromosomes and *crossing-over* between heterozygous genes on the same chromosome. Since we deal with genes on the same chromosome in this paper, we consider modelling only crossing-over. Figure(2.2) shows the way it works. In the figure, one parental genotype ($a.b$) carries all the mutant alleles, and the other parent contains all normal or what are called the wild type alleles ($A.B$). A typical meiotic product (in single crossover : SCO) $a.B$, using the above definition of recombination, is clearly a recombinant, as it is genotypically different from either of the haploid parents $a.b$ and $A.B$. In the figure, we see that different double crossovers (DCO) lead to different allelic combinations in recombinants. The tetrads are classified as parental ditype (PD); denoting no recombinants , Tetrad type (T), denoting equal numbers of recombinants and parental gametes, and non-parental ditype (NPD); denoting all recombinants. These are observable categories by typing gametes (*i.e.*,PD,T and NPD) in asci such as in Figure(2.1).

2.4 Model Assumptions in Comparison to Existing Models

The modeling approach here is quite distinct from that of Lander and Green [39]. Here a detailed model of recombination involves exchanges between 4 chromatids during Prophase

I to generate all possible bivalent configurations in a given interval. The Lander and Green approach is simply based on counting recombination events in a given interval with phase known. As a consequence, in their approach no chromosomal interference is assumed, while the likelihood function developed here does permit and is sensitive to chromosomal interference.

Much work has been carried out on the genetic mapping problem, but most approaches posit some underlying crossover process. For example, Zhao and Speed (see [41], [73], [75]) develop a model in which the crossover process is a stationary renewal process [74]. From this modeling framework they can write down a likelihood specification that leads to the Chi-Square model as a special case [22]. With this likelihood formulation they are able to test the performance of the model for several model systems [73] in a limited way. The limitation of their work is that they assume mathematically tractable processes to describe the crossover process in order to derive analytically tractable likelihood functions for a genetic map. As an example, in their formulation of the Chi-Square model [22] the recombinational intermediates (C) are assumed to be uniformily distributed along a chromosome (i.e., no chromosomal interference), while their resolution is assumed to follow a particular pattern. While the model and hence the likelihood has a parameter m to measure interference, the measure itself is quite abstract and hard to interpret. For example, in their model they state that a non-exchange (C_0) is required to occur m times after each crossover resolution (C_x) followed by a crossover resolution (C_x). Our modeling approach below is distinct by not invoking a particular crossover process to formulate a likelihood function.

Zhao and Speed have calculated very general expressions for the probability of multi-locus recombinants in their Theorem 2.2 in [74], but the limitation of this work is no prescription on how to compute these probabilities except when the assumed process (for example, the Chi-Square model) allows sums over all possible exchanges, reducing to explicit, closed form expressions (Appendix: Theorems 1 and 2 in [73]). Even in this circumstance, they do not present an analysis for more than 10 markers simultaneously. As a consequence it is not clear how their method of likelihood maximization (simplex method) would scale to hundreds of markers without hitting some computational bottleneck. While their modeling and limited likelihood analysis have been illuminating about recombination, they have invoked modeling assumptions about the crossover process that are unnecessary and difficult to verify. In their approach the crossover process is used to connect the observed gametes to the bivalent configurations. In our modeling we begin with the bivalent configurations, sidestepping the specification of a crossover process.

Here we do not make explicit assumptions regarding the underlying crossover process as in earlier work. We develop a probabilistic framework which works directly with all possible bivalent configurations in distinct intervals along the chromosome using the No Chromatid Interference (NCI) model. In this model each possible chromatid exchange between non-sister chromatids in a given interval is equally likely. The term bivalent configuration refers to how chromatids are joined with their non-sister chromatids along the chromosome. In Figure(2.2) all possible bivalent configurations in a tetrad for two markers are depicted. In the next few sections the model is laid out and its connection to the recombination fraction is detailed.

2.5 A Mathematical Formulation for Bivalent Configurations.

For the reader‘s convenience the following glossary of mathematical terms and symbols is presented:

- c_i = probability of a non-sister chromatid exchange for the i^{th} genetic interval
- S_i = sample space for the i^{th} genetic interval
- S^l = sample space for all the genetic intervals
- ϕ_k = k^{th} unique chromatid exchange in S^l
- f_k = k^{th} genotype
- n_j = cell frequency for the j^{th} observed genotype
- $\mathbf{n}$ = vector of cell frequencies n_j.
- R_k = tetrad obtained via equations (2.18) for exchange ϕ_k

- p_j = cell probability for the j^{th} observed genotype
- $N = d_g$ = total number of distinct genotypes (or cells).
- $n = \sum_{j=1}^{d_g} n_j$ = total number of genotypes observed.

Using (2.1) to denote a paternally derived allele and 0 to denote a maternally derived allele, genotypes 11, 00, 10, and 01 in a genetic interval (called, S_i), can result from two simultaneous independent and identical chromatid exchange events (drawn from S). This double chromatid exchange event in S is discrete and not viewed as the product of an underlying continuous crossover process. Chromatid exchanges come in four flavors diagrammed in Figure 2.3. There are a total of 5 kinds of possible chromatid exchanges (including a non-exchange as one of the possibilities). Here our model entertains all possible bivalent configurations (i.e., the specified exchanges on four strands of a bivalent). These exchanges may in principle result from a large number of crossovers occurring in a particular chromosome interval (between markers). We emphasize that our model does not include the crossover(s) per se, but rather a pair of abstract discrete events that are capable of describing all possible bivalent configurations, which could arise from a large number of physical exchanges (i.e., chiasma). Define c_i as the probability of a chromatid exchange in the set S of possible exchanges between any two non-sister chromatids in the i^{th} genetic interval S_i at meiosis. With the assumption of No-Chromatid-Interference (NCI), all chromatid exchanges are equally likely so that:

$$\begin{aligned} S &= \{0,1,2,3,4\} \\ S_i &= S \times S \\ P(i) &= \frac{c_i}{4}; i = 1, \cdots, 4; i \in S \\ P(0) &= 1 - c_i; 0 \in S \end{aligned} \tag{2.1}$$

The element 0 in S denotes the absence of an exchange event. The elements 1, 2, 3, and 4 represent nonsister chromatid exchanges between the pairs of chromosomes, $(1,3)$, $(2,3)$, $(2,4)$, and $(4,1)$. The probability distribution over S_i on possible bivalent configurations between locus A_i and $A_{(i+1)}(i = 1, ..., l-1)$, where l is the total number of loci on the map, can be summarized as follows:

$$P(\{i,j\}) = \frac{c_i^2}{16} I_{\{i \neq 0; j \neq 0\}} + \frac{c_i(1-c_i)}{4} \{I_{\{i=0; j\neq 0\}} + I_{\{i \neq 0; j=0\}}\} + (1-c_i)^2 I_{\{i=j=0\}} \tag{2.2}$$

where, $\{i,j\} \in S_i$.

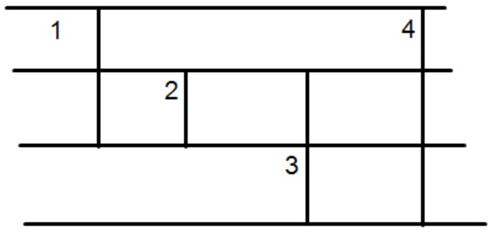


Figure 2.3: Single exchange events are signified by a vertical line, and the exchanges take place between chromatids at the ends of the vertical lines. These 4 strands are found in Prophase-I [19]. All exchanges are equally likely under our hypothesized model. Taken from [63].

2.6 A Probability Model on $S_1 = S \times S$.

The random variable X defined on S_1 is introduced to describe an exchange event in the first interval

$$X = \begin{cases} 0, & \text{no crossover in } S_1; \\ 1, & \text{single crossovers in } S_1; \\ 2, & \text{2-strand double crossovers in } S_1; \\ 3, & \text{3-strand double crossovers in } S_1; \\ 4, & \text{4-strand double crossovers in } S_1. \end{cases}$$

From Eqn. 2.2 we obtain

$$P(X = x) = \begin{cases} (1-c)^2 & \text{if } x = 0 \\ 2c(1-c) & \text{if } x = 1 \\ \frac{c^2}{4} & \text{if } x = 2 \\ \frac{c^2}{2} & \text{if } x = 3 \\ \frac{c^2}{4} & \text{if } x = 4 \end{cases} \tag{2.3}$$

Let $\mathbf{n} = (n_1, n_2, n_3, n_4)'$ be the observed frequency or count vector corresponding to all possible meiotic products for two markers, where one parent is MM and the other parent, OO. In other words these are the counts of resulting genotypes, and the counts in f, are the counts of a multi-locus genotype. With alleles M and O diagnostic of two parents at each of l loci, we have 2^l unique multi-locus genotypes in general. For the case of 2 markers ($l = 2$), we have $2^2 = 4$ genotypes along with their frequencies (or counts) listed below:

- MM with frequency n_1
- MO with frequency n_2
- OM with frequency n_3
- OO with frequency n_4

2.6.1 Conditional Distribution of $\mathbf{n}$ given X(the crossover event)

We derive the conditional distribution of these frequencies $\mathbf{n}$ by considering all of the different meiotic products that might arise from each of the different exchange events in S during meiosis. The exchange event is captured by the unseen random variable X. For example, given $X = 0$, the outcomes are MM and OO, and offspring MO and OM are not seen. Given the random variable X (i.e. the exchange event), the multinomial probabilities associated with the frequencies are the expected Mendelian proportions:

Table 2.1: Conditional Distribution of $\mathbf{n}$ given X

(Different Crossovers)	n_1	n_2	n_3	n_4
0	0.5	0	0	0.5
1	0.25	0.25	0.25	0.25
2	0.5	0	0	0.5
3	0.25	0.25	0.25	0.25
4	0	0.5	0.5	0

Using Eqn. 2.1 and Table 2.1, the model specification and hence the likelihood as a function of the parameter $\Theta = c$ is given by

$$L(\Theta|\mathbf{n}) = \prod_{i=1}^{4} p_i^{n_i} \tag{2.4}$$

where,

$$p_1 = p_4 = \frac{2-2c+c^2}{4} \; ; \; p_2 = p_3 = \frac{2c-c^2}{4}$$

$$p_i = P(n_i) \; ; \; \forall i = 1, \cdots, 4$$

$$\frac{1}{4} \le p_1 \le \frac{1}{2} \; ; \; 0 \le p_2 \le \frac{1}{4}$$

Restrictions on the parameters (see Catchpole and Morgan(1997) [11]) ensure that the recombination probability is less than or equal to $\frac{1}{2}$ see Section2.6.3). The likelihood in Eqn. 2.4 is distinct from [73] because no crossover process is postulated in Eqn. 2.4.

2.6.2 Maximum Likelihood Estimation of the relative crossover frequency c

There are at least two ways to collect data either as random spores or as tetrads in Figure(2.1). The MLE for random spores when two markers are scored is now described to allow comparisions with tetrad data. The following theorem is given without proof in [63].

Theorem 2.1. *"The maximum likelihood estimator of the exchange probability c is unique and is given as follows:*

1. *If $n_1 + n_4 < n_2 + n_3$ then $c_{mle} = 1$*
2. *If $n_1 + n_4 \ge n_2 + n_3$ then c_{mle} is given by the unique solution (in the interval $[0, 1]$) of the following equation:*

$$f(c) = c^2 - 2c + D = 0 \tag{2.5}$$

where,

$$D = \frac{2(n_2+n_3)}{N} \; ; \; N = \sum_{i=1}^{4} n_i$$

"

Proof: The maximum likelihood estimator of c is defined as

$$c_{mle} = \arg \underset{c\in(0,1]}{\text{Max}} \log\left(L\left(c|\mathbf{n}\right)\right)$$

Let the logarithm of the likelihood function be denoted as

$$\mathcal{G}(c) = a\text{log}(2-2c+c^2) + b\text{log}(2c-c^2) \tag{2.6}$$

where,

$$a = n_1 + n_4 \ \text{ and } \ b = n_2 + n_3$$

Now,

$$\begin{aligned} \frac{\partial \mathcal{G}(c)}{\partial c} &= 0 \\ \Rightarrow \quad \frac{a(2c-2)}{2-2c+c^2} - \frac{b(2c-2)}{2c-c^2} &= 0 \\ \Rightarrow \quad (2c-2)\left[a(2c-c^2) - b(2-2c+c^2)\right] &= 0 \quad c \neq 0 \end{aligned}$$

So, the solution set is,

$$\left.\begin{array}{rcl} c &=& 1 \\ a(2c-c^2) - b(2-2c+c^2) &=& 0 \end{array}\right\} \tag{2.7}$$

If the second derivative is calculated,

$$\begin{aligned} \frac{\partial^2 \mathcal{G}(c)}{\partial c^2} &= 2\left[\frac{a}{2-2c+c^2} - \frac{b}{2c-c^2}\right] \\ &\quad - (2c-2)^2\left[\frac{a}{(2-2c+c^2)^2} + \frac{b}{(2c-c^2)^2}\right] \end{aligned}$$

From Eqn. 2.7 upon solving the quadratic equation we find,

$$(a+b)c^2 - 2(a+b)c + 2b = 0$$

$$\Rightarrow \quad c = \frac{2(a+b) \pm \sqrt{4(a+b)^2 - 8(a+b)b}}{2(a+b)} \tag{2.8}$$

So, for $a < b$, roots in Eqn. 2.8 are imaginary pairs, and $c = 1$ is the only solution of (2.7). Using $c = 1$ and the fact that $a < b$ in (2.8) we obtain

$$\frac{\partial^2 \mathcal{G}(c)}{\partial c^2} = a - b < 0$$

Hence $c_{mle} = 1$ for $a < b$ and the claim in part1 of the theorem is proved.

For $a = b$ Eqn. 2.6 becomes

$$\begin{aligned}
\mathcal{G}(c) &= a\left[\log(2-2c+c^2) + \log(2c-c^2)\right] \\
\frac{\partial \mathcal{G}(c)}{\partial c} &= \frac{2a(c-1)^2}{(2-2c+c^2)(2c-c^2)} \\
&> 0 \\
\Rightarrow \quad \mathcal{G}(c) &\uparrow c
\end{aligned}$$

Thus, for $a = b$, $c_{mle} = 1$. Note that $c_{mle} = 1$ is also obtained as the unique solution of Eqn. 2.5 (with $D = 1$) mentioned in the statement of the theorem.

For $a > b$, Eqn. 2.8 has only one root in the interval $(0, 1)$. Also, $c = 1$ is another root which comes from Eqn. 2.7. Now notice that for $c = 1$, Eqn. 2.8 is positive and hence actually corresponds to a minimum rather than maximum. For the other solution, Eqn. 2.8 is clearly negative as the first term goes away because of Eqn. 2.7, and the second term is inherently non-negative. The second term cannot be zero as the solution is not 1.

Hence all the claims of the theorem are proved.

2.6.3 Recombination fraction r and the mapping function:

The recombination fraction (expressed as a percentage) is how distance is measured on a genetic map. The mapping function relates the proportion of recombinants observed, to the underlying physical exchange process captured in the exchange frequency c. Other approaches can be taken. For example, the recombination fraction can be related to the number of crossovers or physical distance We avoid this detailed specification of the crossover process and simple relate the recombination fraction to the relative frequency of exchange c.

The recombination fraction r_i for the i^{th} genetic interval is defined as follows:

$$\begin{aligned}
r_i &= P(\text{at least one meiotic product is recombinant in } S_i) \\
&= c_i\left(1 - \frac{c_i}{2}\right) \\
&= \phi(c_i) \qquad (2.9)
\end{aligned}$$

Also,

$$\underset{c_i \in [0,1]}{\text{Max}}\, r = \underset{c_i \in [0,1]}{\text{Max}}\, \phi(c_i) = \frac{1}{2}, \quad \forall i = 1, \cdots, l-1,$$

which is the theoretical maximum of the recombination fraction from the theory of Mendelian genetics.

2.7 Likelihood of Tetrad Types

In the case of two markers MLEs for tetrad analysis are developed to make contact with results in the literature: heuristic estimators of map distances need standard errors, and to provide a reasonable initialization for recombination distances, when many markers are followed simultaneously.

Often in tetrad analysis we have additional information on the gametes produced.

Let T be a random variable defined as follows:

$$T = \begin{cases} 1 & \text{if } x = 0, 2 \\ 2 & \text{if } x = 1, 3 \\ 3 & \text{if } x = 4 \end{cases} \tag{2.10}$$

where,

$t = 1$ is known as the Parental ditype(PD)

$t = 2$ is known as the Tetrad type(T)

$t = 3$ is known as the Non-Parental ditype(NPD)

The values of T designate the spore package makeup or tetrad type [19]. Figure 2.2 depicts a model of the sample space for T.

From Eqn. 2.3, the likelihood of $\mathbf{t} = (t_1, t_2, t_3)'$ as a function of c is

$$L(c|\mathbf{t}) = \prod_{i=1}^{3} p_{T_i}^{t_i} \tag{2.11}$$

where,

$$\begin{aligned}
t_1 = &\quad \text{frequency of PD} \\
t_2 = &\quad \text{frequency of T} \\
t_3 = &\quad \text{frequency of NPD} \\
p_{T_1} = &\quad (1-c)^2 + \tfrac{c^2}{4} \\
p_{T_2} = &\quad 2c(1-c) + \tfrac{c^2}{2} \\
p_{T_3} = &\quad \tfrac{c^2}{4}
\end{aligned}$$

We also consider another widely used heuristic estimator of r_{heu} [19] under the model specified by (2.10) as

$$r_{heu} = \frac{0.5t_2 + 3t_3}{N_t} \ ; \ N_t = \sum_{i=1}^{3} t_i$$

2.7.1 Maximum Likelihood Estimation of $c_{mle.td}$:

Theorem 2.2. *The maximum likelihood estimator of the exchange probability $c_{mle.td}$ under the tetrad model as specified by Eqn. 2.11 always exists and is unique under some regularity conditions and is given by :*

$$c_{mle.td} = \arg \underset{\mathcal{A}}{\mathrm{Max}} \left\{ \log\mathrm{L}\left(\mathcal{A}\right) \right\} \tag{2.12}$$

where,

L is given by Eqn. 2.11 and $\mathcal{A}$ is the set of all real roots in the interval $(0, 1]$ of the equation below (including 1 if not present in the interval).

$$a_3c^3 + a_2c^2 + a_1c + a_0 = 0 \tag{2.13}$$

where,

$$\begin{aligned}
a_3 &= -30T_1 - 30T_2 - 30T_3 \\
a_2 &= 64T_1 + 68T_2 + 88T_3 \\
a_1 &= -32T_1 - 56T_2 - 88T_3 \\
a_0 &= 16T_2 + 32T_3
\end{aligned}$$

The Regularity Conditions are:

1. $2T_1 - 2T_2 + 2T_3 < 0$ *or*

2. $T_1 > T_2 > T_3$ *and* $\mathcal{G}(\mathbf{T}) < 0$

[for an expression for $\mathcal{G}(\mathbf{T})$*see the proof of Lemma2.2 in the appendix.]*

Proof: Differentiating the log-likelihood function given by Eqn. 2.11 we arrive at the estimating Eqn. 2.13. Under the first regularity condition using *Descartes* rule of signs [13], a solution in the interval $(0, 1]$ is guaranteed. Under the second regularity condition using the lemma below all the roots of the estimating equation are real. In the lemma we will also see that one solution always lies in the interval $(1, 2]$ which is not considered directly as an MLE since the parameter space is $(0, 1]$. The MLE is found using Eqn. 2.12. The reason for including 1 in the set is that sometimes the global maximizer of L is in the interval $(1, 2]$ and the function keeps increasing after the other two local extrema, and a check is necessary to see if the likelihood value at 1 is greater.

Lemma 2.1. *Under the condition*

$$T_1 > T_2 > T_3 \ \ \textit{and} \ \ \mathcal{G}(\mathbf{T}) < 0$$

the roots of the Eqn. 2.13 are all real.

Proof: See the Appendix.

2.8 Identifiability of the single spore and tetrad model

Silvey ([59],Ch.4) defines a model to be identifiable if no two values of the parameters give the same probability distribution of the data. In the single spore model described by equation(2.4) there are only 2 independent observable parameters namely, p_1 and p_2. Their model equations are

$$\begin{aligned} 2 - 2c + c^2 &= 4p_1 \\ 2c - c^2 &= 4p_2 \end{aligned}$$

We see that both the model equations are monotonic in the crossover probability c in the interval $[0, 1]$, and hence the model is identifiable. For the tetrad model described by equation(2.11) the model equation corresponding to tetrad NPD is clearly monotonic in c and hence the tetrad model is identifiable too.

2.9 Data Analysis

Giles *et al.* [25] obtained tetrads on several biochemical mutants to understand recombination and chromosome inteference. The tetrad counts for the loci *hist-2, nic-2, al-2, ad-3A* and *ad-3B* on 646 complete asci are reported in Table2.2.

Table 2.2: Summary of tetrad analyses of 646 complete asci

Ascus Classification	***hist-2-nic-2***	***hist-2-al-2***	***nic-2-al-2***	***ad-3A-ad-3B***
Parental ditype	538	115	125	640
Tetratype	108	486	487	6
Nonparental ditype	0	45	34	0

It is important to point out the difference in the nature of information between single spores and tetrad data. In the single spore model one spore is taken from each ascus ideally, from one diploid individual. In contrast spores inside a complete ascus in tetrad data are not independent because they are gametes from the same diploid parents. Although both single spores and tetrad data reflect on the underlying crossover events, tetrad data do so more directly. For example, an NPD tetrad directly indicates a four strand double crossover. Since we model the underlying recombination process which is fundamental to any observed cross, it is possible to ascribe a probability distribution on the tetrad types directly from the recombination process. Note that though an NPD tetrad ensures 50% f_2 and 50% f_3 (see Section2.6.1) it would be wrong to calculate the probability of an NPD tetrad via the conditional distribution in Section2.6.1 as the spores in the NPD asci are not independent. We compute the recombination fraction r for the heuristic estimator r_{heu} and maximum likelihood estimator $r_{mle.td}$ under the model described in equation(2.11) and for the maximum likelihood estimator r_{mle} under the single spore model described in equation(2.4).The results are tabulated in Table2.3. All the estimators are observed to agree quite well.

Table 2.3: Test Results on the Data set:

Ascus Classification	$\mathbf{r_{mle}}$	$\mathbf{r_{mle.td}}$	$\mathbf{r_{heu}}$
hist-2-nic-2	0.083591571	0.083630811	0.0835
hist-2-al-2	0.445820264	0.435380594	0.4458
nic-2-al-2	0.42956649	0.422590286	0.4295
ad-3A-ad-3B	0.004644165	0.004643966	0.0046

where,

$$\begin{aligned} r_{mle} &= \text{MLE estimate of the recombination fraction under the single spore model} \\ r_{mle.td} &= \text{MLE estimate of the recombination fraction under the tetrad model} \\ r_{heu} &= \text{heuristic estimator of the recombination fraction } r \\ &= \frac{(0.5T + NPD)}{PD + T + NPD} \end{aligned}$$

Any of these estimators would provide a good initialization for iterative computation of the MLE for a single spore model in the case of multiple markers (see Section 2.11).

As all the estimators perform well, it becomes interesting to study their accuracy. In the following section we compare the efficiency of these three estimators.

2.10 Comparison of Efficiency among r_{mle}, $r_{mle.td}$ and r_{heu}

Theorem 2.3. *The estimator of the recombination fraction based on the tetrad model is uniformly more efficient than that of the single-spore model. This shows that tetrad data does have more information about the recombination fraction than the single spore data.*

Proof: Variances of the maximum likelihood estimators $r_{mle.td}$ and r_{mle} are obtained via the Fisher-information [54]. The variance of the heuristic estimator r_{heu} is obtained in a straightforward manner using tetrad probabilities given by equation(2.11). The details of the derivation are included in the appendix. The plot (Fig.2.4) compares the variances of r_{mle},$r_{mle.td}$ and r_{heu} using equations (2.14), (2.15) and (2.16).

$$\sigma^2_{r_{mle.td}} = \frac{(1-c)^2}{(\sum_{i=1}^{3} t_i)\left[\frac{1}{2} + \frac{50c^2 - 80c + 24}{16(1-c)^2 + c^2} + \frac{18c^2 - 24c + 16}{2c^2 + 8c(1-c)}\right]} \tag{2.14}$$

$$\sigma^2_{r_{heu}} = \frac{-\frac{c^4}{4} + c^3 - \frac{9}{8}c^2 + \frac{c}{2}}{(\sum_{i=1}^{3} t_i)} \tag{2.15}$$

$$\sigma^2_{r_{mle}} = \frac{(2 - 2c + c^2)(2c - c^2)}{4(\sum_{i=1}^{4} n_i)} \tag{2.16}$$

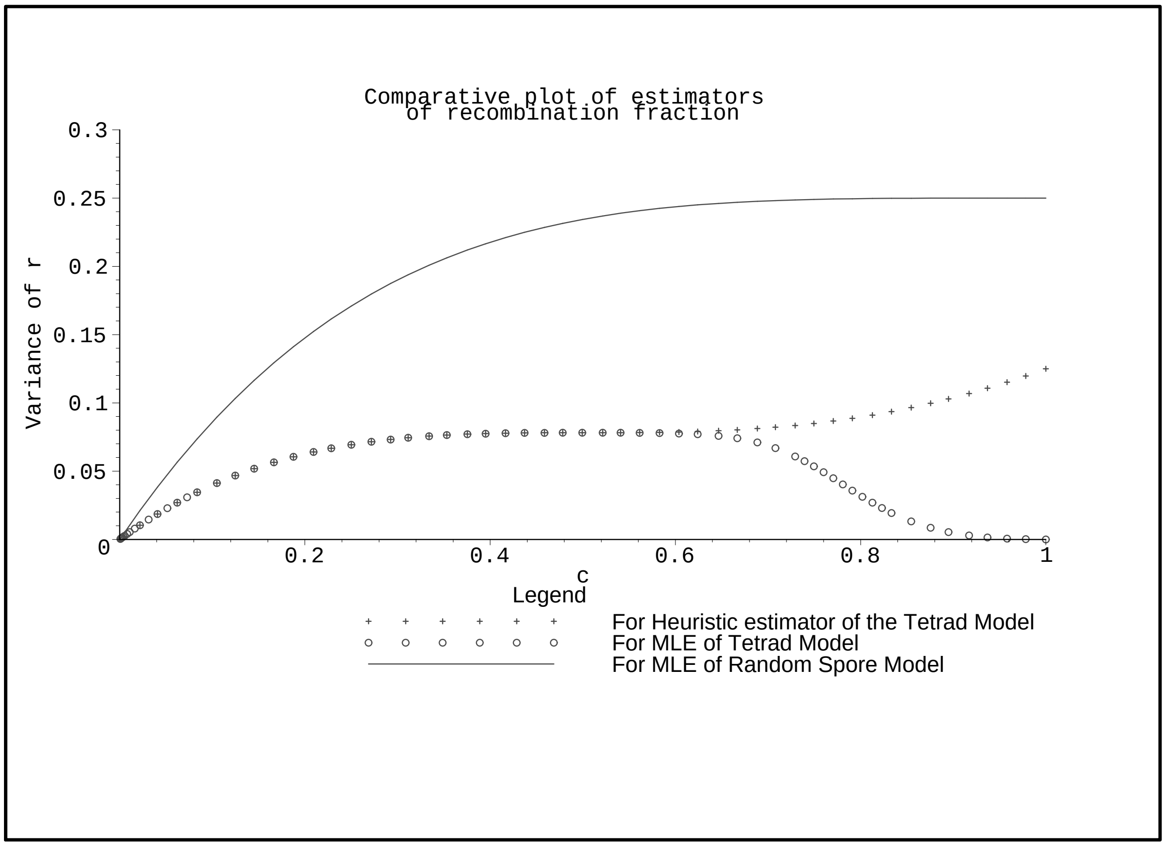


Figure 2.4: Comparative plot of variances r_{mle}, $r_{mle.td}$ and $r_{heu.td}$.

These standard errors have not been available before for the fungal genetic community. In the plot (Fig.2.4) we see that the estimators $r_{mle.td}$ and r_{heu} of the tetrad model are uniformly more efficient than r_{mle} of the single spore model. Although this was intuitively expected, it was not quite obvious. Although identifying a tetrad type involves extracting more information (hence intuitively smaller variance) than for a single spore, we had different probability distributions for them, and a check was necessary to confirm intuition. In the next section we expand on the single spore model to account for multiple markers. It would appear that the amount of information in tetrads is usually at least a factor of 4 greater than that of single random spore. If obtaining tetrads is not more than 4 times the work of random spores, then tetrads are worth obtaining versus random spore data. This is intuitive because each tetrad in Figure(2.1) allows us to observe potentially up to 4 distinct recombination events. The only situation where the relative efficiency approaches 1 is as the recombination fraction approaches zero.

2.11 Multi locus Model for Random-Spore Data

Let ϕ_k denote a unique chromatid exchange on S^l as described below:

$$\phi_k = i_1 \times i_2 \times ... \times i_{l-1} \tag{2.17}$$

where,

$$k = i_1.i_2.i_3...i_{l-1} \ ; \ i_j \in S_i \ ; \ \phi_k \in S^l = \prod_{i=1}^{l-1} S_i$$

From this point on, for the sake of brevity, we may abbreviate term *chromatid exchanges* to simply *exchanges* when referring to ϕ_k.

Let f_k denote a multi-locus genotype with l loci:

$$f_k = i_1 \times i_2 \times ... \times i_{l-1} \times i_l$$

where,

$$k = i_1.i_2.i_3...i_l \ \ ; \ \ i_j = 0, 1; \ \ \forall j = 1, \cdots, l$$

The indices $i_j = 1$ and $i_j = 0$ indicate the paternal and maternal alleles respectively. The progeny are obtained by exchanges between homogeneous parents. The observed data set can be represented as:

$$\mathbf{n} = \left\{ n_j \ \ ; \ \ \forall j = 1, \cdots, 2^l \right\}$$

where, n_j is the observed frequency of f_j.

2.11.1 Probability distribution on S^l

Let us define the following functions

$$\begin{aligned} f^0(a) &= (a_1, a_2, a_3, a_4)' \\ f^1(a) &= (a_3, a_2, a_1, a_4)' \\ f^2(a) &= (a_1, a_3, a_2, a_4)' \\ f^3(a) &= (a_1, a_4, a_3, a_2)' \\ f^4(a) &= (a_4, a_2, a_3, a_1)' \\ \text{where,} & \\ a &= (a_1, a_2, a_3, a_4)' \\ a_i &= 0, 1 \;\; \forall i \end{aligned} \tag{2.18}$$

Equation (2.18) represents the elements of set S (see equation (2.1)) as mathematical functions. For example, the function $f^0(a)$ represents element 0, showing no chromatid exchange, whereas function $f^4(a)$ represents element 4, indicating that the first and fourth strand have had an exchange.

The function $f_{ij}(a) = f_j(f_i(a))$ corresponds to events in S_i accounting for all possible bivalent configurations. For a particular chromatid exchange ϕ_k a model tetrad can be generated at meiosis using the function f_{ij}. The matrix R_k of size $4 \times l$ defines a simulated tetrad below [57]:

$$R_k = \left(R_0 R_1 \cdots R_{(l-1)}\right) \tag{2.19}$$

where,

$$R_0 = (1100)' \;\; ; \;\; R_i = f_{jk}(R_{i-1}) \, \forall i = 1, \cdots, l-1$$

and the i^{th} genetic interval S_i contains the observed chromatid exchange $\{j, k\}$. That is, R_k possesses 4 rows which correspond to the 4 gametes in a tetrad during meiosis if the chromatid exchange ϕ_k had occurred according to our model.

The conditional distribution of f_i for a given ϕ_k is calculated as

$$P(f_i|\phi_k) = \frac{1}{4} \sum_{j=1}^{4} I_{f_{i \in R_{k(j,.)}}} \tag{2.20}$$

where, $R_k(j,.)$is the j^{th} row of R_k.

The marginal density of a single spore f_i is given by

$$\begin{aligned} P(f_i) &= \sum_k P(f_i|\phi_k) \times P_k \\ &= C \times P \end{aligned} \tag{2.21}$$

where, C is the conditional probability matrix defined by :

$$\left.\begin{aligned} C &= ((c_{ki})) \\ c_{ki} &= P(f_i|\phi_k) \quad \text{(from equation (2.20))} \end{aligned}\right\} \tag{2.22}$$

and P is given by,

$$\begin{aligned} P &= (P_k;\ \ \forall k)' \\ P_k &= P(\phi_k) \\ &= \prod_{j=1}^{l-1} P\left(I_j = i_{k,j}\right) \end{aligned} \tag{2.23}$$

where, $i_{k,j} \in S_j$ and the probability distribution $P\left(I_j = i_{k,j}\right)$ is as defined in equation (2.2).

Let $\Theta = (c_1, c_2, ..., c_{l-1})'$ denote the unknown parameter vector in the model. The likelihood of $\mathbf{n}$ viewed as a function of Θ is specified as :

$$\begin{aligned} L(\Theta|\mathbf{n}) &\propto \prod_{j=1}^{N} P(f_i)^{n_j} \\ &= \prod_{j=1}^{N} \left[\sum_k \left[P\left(f_i|\phi(k)\right) \times \prod_{j=1}^{l-1} P_j\left(I_{k,j} = i_{k,j}\right)\right]\right]^{n_j} \end{aligned} \tag{2.24}$$

The log-likelihood function is then :

$$\ln(\Theta|\mathbf{n}) \propto \sum_{j=1}^{N} n_j \log\left[\sum_k \left[\frac{1}{4}\sum_{j=1}^{4} I_{\{f_i \in R_k(j,.)\}} \prod_{j=1}^{l-1} P_j\left(I_{k,j} = i_{k,j}\right)\right]\right] \tag{2.25}$$

2.11.2 Computation of MLE of the Crossover Probability Vector Θ via the EM Algorithm

Lander and Green [39] suggested the application of the EM algorithm to the genetic mapping problem. Consider two sample spaces $\mathcal{X}$ and $\mathcal{N}$ and a many $\rightarrow$ one mapping from $\mathcal{X}$ to $\mathcal{N}$. The actual data $\mathbf{n} = (n_j; j = 1, \cdots, 2^l)$ are a realization of $\mathcal{N}$. Let X_{kj} be the random variable corresponding to X_j which denotes the frequency of events like ϕ_k for random spores f_j. Define

$$X_{.j} = \sum_k X_{kj} \quad \text{and} \quad X_{k.} = \sum_j X_{kj}$$
$$X = \sum_k X_{k.} = \sum_j X_{.j}$$

where,

$$X_{kj}\Big|(X = N) \overset{\text{dist}}{=} X_{kj}\Big|(X_{.j} = n_j) \sim B\left(n_j, \pi_{k|j}\right)$$

We refer to $\mathcal{X}$ as complete data and $\mathcal{N}$, as incomplete data.

The log-likelihood with complete data is given by :

$$l^c(\Theta, x) = \sum_k x_{k.}\log(\pi_k) \tag{2.26}$$

where,

$$\begin{aligned} \Pi &= (\pi_1, \pi_2, ..., \pi_K)' \\ \pi_k &= \sum_j \pi_{k|j} \\ \pi_k &= f : [0,1]^{l-1} \rightarrow [0,1] \quad \text{and is defined in Eqn. 2.23.} \end{aligned}$$

The log-likelihood with incomplete data is given by :

$$l(\Theta; \mathbf{n}) = \sum_j n_j \log(p_j)$$

where,

$$\begin{aligned} p_j &= P\left(\text{observing a single spore of } f_j\right) \\ &= P\left(f_j\right), \quad \text{a function of } \Theta \text{ and is defined in Eqn. 2.21.} \end{aligned}$$

In the EM algorithm [20] we do not maximize $l(\Theta; \mathbf{n})$ directly to obtain the ML estimates of Θ, but iteratively maximize $l^c(\Theta, x)$, averaged over all possible values of the complete data, given the incomplete data. That is, the objective function is defined as

$$Q(\Theta|\Theta^{(h)}) = E\left[l^c(\Theta; x|\mathbf{n}, \Theta^{(h)}\right] \quad ,$$

and we iteratively maximize $Q\left(\Theta\middle|\Theta^{(h)}\right)$ i.e.,

$$\Theta^{(h+1)} = \arg\underset{\Theta}{\text{Max}}\, Q\left(\Theta|\Theta^{(h)}\right)$$

E-Step:

$$\begin{aligned} Q\left(\Theta|\Theta^{(h)}\right) &= E\left[\sum_k x_k \log(\pi_k)\Big|\mathbf{n},\Theta^{(h)}\right] \\ &= \sum_k \left[\log(\pi_k)E(x_k\Big|\mathbf{n},\Theta^{(h)}\right] \\ &= \sum_k \left[\log(\pi_k)\sum_j n_j\pi_{k|j}(\Pi^{(h)})\right] \\ &= \sum_k \left[\log(\pi_k)n_k^{(h)}\right] \text{ where } n_k^{(h)} = \sum_j n_j \pi_{k|j}^{(h)} \end{aligned} \tag{2.27}$$

Note that

$$\pi_{k|j}^{(h)} = P(x_{kj}=1|f_j) = \frac{\pi_{j|k}^{(h)} \times \pi_k^{(h)}}{p_j^{(h)}} \tag{2.28}$$

M-Step:

Using Eqn. 2.27 above we find,

$$\begin{aligned} Q\left(\Theta|\Theta^{(h)}\right) &= \sum_k \left(\sum_{m=1}^{l-1}\log\left(P\left(I_m=i_{k,m}\right)\right)\right)n_k^{(h)} \\ &= \sum_{m=1}^{l-1}\left(\sum_k \log\left(P\left(I_m=i_{k,m}\right)\right)n_k^{(h)}\right) \\ &= \sum_{m=1}^{l-1}\left\{N_{0,m}\log\left(1-c_m^2\right)+N_{1,m}\left(\log\left(c_m(1-c_m)\right)\right)+2N_{2,m}\log(c_m)\right\} \\ &= \sum_{m=1}^{l-1}\left\{\log(1-c_m)\left[2N_{0,m}+N_{1,m}\right]+\log(c_m)\left[2N_{2,m}+N_{1,m}\right]\right\} \end{aligned}$$

Now

$$\begin{aligned} \frac{\partial Q\left(\Theta|\Theta^{(h)}\right)}{\partial\Theta} &= 0 \text{ for } m=1,\cdots,l-1 \\ \Rightarrow \frac{c_m}{1-c_m} &= \frac{2N_{2,m}+N_{1,m}}{2N_{0,m}+N_{1,m}} \\ \Rightarrow c_m &= \frac{2N_{2,m}+N_{1,m}}{2N_m} \end{aligned} \tag{2.29}$$

where

$$N_m = \sum_k n_k^{(h)} = N_{0,m}+N_{1,m}+N_{2,m}$$

$$N_{0,m} = \sum_{k\big|i_{k,m}=(0,0)} n_k^{(h)}$$

$$N_{1,m} = \sum_{\substack{k \mid i_{k,m}=(i_1,i_2) \\ i_1=0 \text{ (Strict)OR } i_2=0}} n_k^{(h)}$$

$$N_{2,m} = \sum_{\substack{k \mid i_{k,m}=(i_1,i_2) \\ i_1\neq 0 \text{ AND } i_2\neq 0}} n_k^{(h)} \text{ and so on.}$$

Note that, $i_{k,m}$ denotes an event in S_m for the crossover ϕ_k.

Thus,

$$\begin{aligned}\Theta^{(h+1)} &= \left(c_m^{(h+1)} \ \forall m = 1, \cdots, l-1\right)' \\ \text{where } c_m^{(h+1)} &= \left(\frac{2N_{2,m} + N_{1,m}}{2N_m}\right)^{(h)}\end{aligned} \tag{2.30}$$

Computational implementation of the EM algorithm is described in a separate paper [63].

2.12 Computing the Standard Error of the MLE

We employ the *SEM* algorithm [44] to compute the standard errors of $\Theta = (c_m, m = 1, \cdots, l-1)$. Its description paraphrases the description in [63]. The large sample variance-covariance matrix is calculated by (Equation (2.3.5) [44])

$$V = I_{oc}^{-1} + \Delta V \tag{2.31}$$

where,

$$\Delta V = I_{oc}^{-1} D (I - D)^{-1}$$

and D is the matrix determining the rate of convergence of EM and I_{oc} is defined as below:

$$I_{oc} = E\left[I_o\left(\theta|Y\right)|Y_{obs}, \theta\right]\Big|_{\theta=\theta^*} \tag{2.32}$$

where, $I_o(\theta|Y)$ is the complete-data observed information matrix.

The EM algorithm described in section2.11.2 implicitly defines a mapping $\theta \to M(\theta)$ by equation(2.29) from the parameter space of θ,$(0, 1]^{l-1}$, to itself such that

$$\theta^{(t+1)} = M(\theta^{(t)}), \qquad \text{for } t = 0, 1, \cdots$$

Since $M(\theta)$ is continuous and $\theta^{(t)}$ converges to the MLE θ^*(using EM algorithm), then θ^* is a root of

$$\theta^* = M(\theta^*)$$

Therefore in the neighborhood of θ^*, by a Taylor series expansion, we obtain

$$\theta^{(t+1)} - \theta^* \approx (\theta^{(t)} - \theta^*)D$$

where,

$$D = \left(\frac{\partial M_j(\theta)}{\partial \theta_i}\right)\Big|_{\theta=\theta^*}$$

is the $(l-1)\times(l-1)$ jacobian matrix for $M(\theta) = (M_1(\theta), \cdots, M_{l-1}(\theta))$ evaluated at $\theta = \theta^*$.

2.12.1 Computation of D

Define d_{ij} to be the $(i,j)^{th}$ element of D and define $\theta^{(t)}(i)$ to be

$$\theta^{(t)}(i) = \left(\theta_1^*, \cdots, \theta_i^{(t)}, \theta_{i+1}^*, \cdots, \theta_{l-1}^*\right) \tag{2.33}$$

That is, only the i^{th} component in $\theta^{(t)}(i)$ is active since the other components are fixed at their MLE's. By the definition of d_{ij}, we have

$$\begin{aligned} d_{ij} &= \frac{\partial M_j(\theta^*)}{\partial \theta_i} \\ &= \lim_{\theta_i \to \theta_i^*} \frac{M_j\left(\theta_1^*, \cdots, \theta_{i-1}^*, \theta_i, \theta_{i+1}^*, \cdots, \theta_{l-1}^*\right) - M_j\left(\theta^*\right)}{\theta_i - \theta_i^*} \\ &= \lim_{\theta_i \to \theta_i^*} \frac{M_j\left(\theta^{(t)}(i)\right) - \theta_j^*}{\theta_i - \theta_i^*} \\ &= \lim_{t\to\infty} d_{ij}^{(t)} \end{aligned}$$

The following steps are performed to compute the d_{ij}'s.

INPUT: θ^* and $\theta^{(t)}$.

Step 1. Run the usual E and M steps to obtain $\theta^{(t+1)}(i)$.

Repeat steps 2-3 for $i = 1, \cdots, l-1$.

Step 2. Calculate $\theta^{(t)}(i)$ from Equation(2.33), and treating it as the current estimate of

θ, perform one iteration of EM¡ to obtain $\theta^{\,(t+1)}(i)$.

Step 3. Obtain the ratio

$$d_{ij} = \frac{\theta_j^{\,(t+1)}(i) - \theta_j^*}{\theta_i - \theta_i^*}, \qquad \text{for } j = 1, \cdots, l-1$$

OUTPUT: $\theta^{(t+1)}$ and $\{d_{ij}^{(t)}, i, j = 1, \cdots, l-1\}$.

We obtain $d_{ij}$when the sequence $d_{ij}^{(t^*)}, d_{ij}^{(t^*+1)}, \cdots$ is stable for some t^*. This process may result in using different values of t^* for different d_{ij} elements.

2.12.2 Evaluation of I_{oc}^{-1}.

The complete-data information for the $(i, j)^th$ element ($i = 1, \cdots, l-1$ and $j = 1, \cdots, l-1$) is given by

$$\begin{aligned} I^o(i,j) &= -\frac{\partial^2 f^c(x,\theta)}{\partial\theta_i \partial\theta_j} \\ &= -\sum_k x_k \frac{\pi_k \frac{\partial^2}{\partial\theta_i\partial\theta_j}(\pi_k) - \frac{\partial}{\theta_i}(\pi_k)\frac{\partial}{\theta_j}(\pi_k)}{\pi_k^2} \end{aligned}$$

The complete-data information for the $(i, j)^{th}$ element ($i = 1, \cdots, l-1$ and $j = 1, \cdots, l-1$) is

$$\begin{aligned} I^o(i,j) &= -\frac{\partial^2 f^c(x,\theta)}{\partial\theta_i \partial\theta_j} \\ &= -\sum_k x_k \frac{\pi_k \frac{\partial^2}{\partial\theta_i\partial\theta_j}(\pi_k) - \frac{\partial}{\theta_i}(\pi_k)\frac{\partial}{\theta_j}(\pi_k)}{\pi_k^2} \end{aligned}$$

Using Equation(2.32), we obtain

$$\begin{aligned} I_{oc} &= E\left[I^o(\theta|Y)\,|Y_{obs}, \theta\right]\Big|_{\theta=\theta^*} \\ &= \sum_k \left[-x_k \frac{\pi_k \frac{\partial^2}{\partial\theta_i\partial\theta_j}(\pi_k) - \frac{\partial}{\partial\theta_i}(\pi_k)\frac{\partial}{\partial\theta_j}(\pi_k)}{\pi_k^2}\left[E\left(x_k \Big| n, \theta^*\right)\right]\right] \\ &= \sum_k \left[-x_k \frac{\pi_k \frac{\partial^2}{\partial\theta_i\partial\theta_j}(\pi_k) - \frac{\partial}{\partial\theta_i}(\pi_k)\frac{\partial}{\partial\theta_j}(\pi_k)}{\pi_k^2}\left[\sum_{j'=1}^{N} n_j' \pi_{k|j'}(\theta^*)\right]\right] \\ &= \sum_{j'=1}^{N} \frac{n_{j'}}{p_{j'}}\left[\sum_k \left[\frac{\pi_{j'|k}}{\pi_k}\frac{\partial}{\partial\theta_i}(\pi_k)\frac{\partial}{\partial\theta_j}(\pi_k) - \pi_{j'|k}\frac{\partial^2}{\partial\theta_i\partial\theta_j}(\pi_k)\right]\right] \quad \text{using Equation(2.28)} \end{aligned}$$

2.12.3 Empirical Validation of Standard Errors

We performed a study to verify empirically equation(2.31). We used the first 9 genes of chromosome 7 of *Neurospora crassa* [46] for the study. First, we computed the parameter vector θ using equation(2.29), and we used the estimated parameter to generate 50 data sets from the model described in equation(2.1). For each model we computed the parameter vector θ and calculate the empirical standard error for all the data sets. The following table shows the theoretical and empirical estimates of the parameter vector θ.

Table 2.4: Empirical Validation of Standard Errors:

Genetic Intervals	**Theoretical Estimates**	**Empirical Estimates**
Tel VIIL - AP8e.1	0.0682	0.0733
AP8e.1 - 5:5A	0.0935	0.2225
5:5A - 00003	0.0599	0.0273
00003 - ccg-9	0.1012	0.0843
ccg-9 - pho-4	0.0902	0.1712
pho-4 - nic-3	0.1036	0.1690
nic-3 - AP12i.2	0.1415	0.1417
AP12i.2 - AP11c.3	0.1307	0.0475

2.13 Computing Goodness-of-Fit of the Model

Let $\mathbf{R_i}(i = 1, 2, \cdots, l-1)$ be a multinomial vector for the i^{th} genetic interval, where

$$\mathbf{R_i} = (R_{i1}, R_{i2}, R_{i3}, R_{i4})' \ ; \ R_{ij} = \sum_{k=1}^{n} R_{ijk} \ ; \ R_{ijk} \sim M(1, P_{ij}(\theta)) \ ; \ j = 1, \cdots, 4 \tag{2.34}$$

In other words, R_{ij}= count of genotype f_j as in equation(2.4) for the i^{th} genetic interval. Assuming independence of the multinomial counts by virtue of the assumption of independent DNA breakage across the intervals, we get

$$\mathbf{R_i} \overset{ind}{\sim} M(n, \mathbf{P_i}(\theta)) \ \ \forall i = 1, 2, \cdots, l-1$$

where, $\mathbf{P_i}(\theta) = (P_{i1}(\theta), \cdots, P_{i4}(\theta))'$ and $\theta = (c_1, c_2, \cdots, c_{l-1})'$

This independence assumption is an approximation, but a good one (See Section(2.15)). We use the standard Pearson chi-squared statistic as a measure of goodness-of-fit for the model proposed as in Equation(2.1) and is defined as

$$\mathcal{X}^2 = \sum_{i=1}^{l-1} \frac{\left(O_i - nP(\hat{\theta})\right)^2}{nP(\hat{\theta})}$$

which asymptotically follows a central chi-squared distribution with $n - l$ degrees of freedom [1]. The following lemma and its proof are repeated from [63] for the reader's convenience.

Lemma 2.2. *"Under the model described by Equation(2.1) at any particular locus only one of the tetrad patterns 1 1 0 0, 0 1 1 0, 1 0 1 0, 1 0 0 1, 0 1 0 1 and 0 0 1 1 could occur.*

Proof: Recall that a tetrad pattern is an arrangement of the alleles of a particular gene in the simulated tetrad generated by a given crossover under the model described by Equation(2.1), where 1 and 0 indicate the parental alleles for the particular locus. Let P_1,P_2,$\cdots$,P_6 denote the tetrad patterns 1 1 0 0, 0 1 1 0, 1 0 1 0, 1 0 0 1, 0 1 0 1 and 0 0 1 1 respectively. The tetrad pattern at locus i for a crossover value s_i in the i^{th} genetic interval S_i is given by

$$T_{s_i} = f_k(f_j(T_{s_{i-1}}))$$

where, $s_i = \{j, k\} \in S_i$ in equation(2.2) and $f_k(.)$ and $f_j(.)$ are obtained from equation(2.18). Note that $T_{s_0} = P_1$. In order to generate the patterns beginning with pattern T_{s_0} along with their sample points (Table2.5), we can see that all the sample points with the source pattern $T_{s_0} = P_1$ correspond to the patterns within $P_i(i = 1, \cdots, 6)$. Next we enumerate tetrad patterns beginning with source patterns $P_i(i = 1, 2, \cdots, 6)$ and from Table2.5 it is clearly seen that tetrad patterns cannot lie outside the set $P_i(i = 1, \cdots, 6)$. Hence the lemma is proved."

The probability distributions of the tetrad patterns are given below:

Next we derive probability distribution of the multinomial counts for each genetic interval and for each of the beginning strand among the possible tetrad patterns.

Theorem 2.4. *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ define two classes of tetrad patterns as below:*

$$\mathcal{C}_1 = \{P_1, P_6\} \qquad \mathcal{C}_2 = \{P_2, P_3, P_4, P_5\}$$

The probability distribution of $\mathbf{R_i}$ *for a tetrad pattern* $P_l(l = 1, \cdots, 6)$ *as the beginning strand is given by:*

$$P\left(R_{ij} \middle| P_l \in \mathcal{C}_1\right) = \begin{cases} \frac{c^2-2c+2}{4} & \text{if } R_{ij} = f_1, f_4 \\ \frac{2c-c^2}{4} & \text{if } R_{ij} = f_2, f_3 \end{cases} \tag{2.35}$$

$$P\left(R_{ij} \middle| P_l \in \mathcal{C}_2\right) = \begin{cases} \frac{c^2-4c+8}{16} & \text{if } R_{ij} = f_1, f_4 \\ \frac{4c-c^2}{16} & \text{if } R_{ij} = f_2, f_3 \end{cases} \tag{2.36}$$

Now,from equation(2.34)

$$\begin{aligned} P_{ij}(\theta) &= E\left[R_{ijk}\right] \\ &= E_{G_i}\left[E\left[R_{ijk} \middle| G_i\right]\right] \\ &= \sum_{k=1}^{2} P_{G_i}\left(\mathcal{C}_k\right) P\left(R_{ij} \middle| P_l \in \mathcal{C}_k\right) \end{aligned} \tag{2.37}$$

where, G_i is the random variable denoting a strand pattern among $P_1, \cdots, P_6$. Note that the distribution of G_i is of branching type where i^{th} genetic interval corresponds to what we call the i^{th} generation. Since the genetic map starts with the pattern P_1, the distribution of G_1 *i.e.*,the 1^{st} generation, is given by the first row in Table 2.6. We see that

$$P_{G_i}\left(\mathcal{C}_1\right) = \sum_{l=1,6} P_{G_i}\left(P_l\right)$$

and

$$P_{G_i}\left(\mathcal{C}_2\right) = \sum_{l=2}^{5} P_{G_i}\left(P_l\right)$$

Hence in order to compute Equation(2.37) we have to find the probability distribution of the i^{th} generation random variable G_i *i.e.*, compute $P_{G_i}\left(P_l\right)\ \forall i = 1, \cdots, l-1$ and $\forall l = 1, \cdots, 6$. The following theorem provides a set of recurrence relations for obtaining them.

Theorem 2.5. *The set of recurrence relations for computing Equation(2.37) is given as follows:*

$$\begin{aligned} P_{G_1}\left(P_1\right) &= (1-c_1)^2 + \frac{c_1^2}{4} \\ P_{G_1}\left(P_6\right) &= \frac{c_1^2}{4} \\ P_{G_1}\left(P_l\right) &= \frac{c_1}{2}(1-c_1) + \frac{c_1^2}{8} \quad \forall\, l = 2, \cdots, 5 \end{aligned}$$

For $i = 2, \cdots, l-1$

$$\begin{aligned}
P_{G_i}(P_1) &= \left((1-c_i)^2 + \frac{c_i^2}{4}\right) P_{G_{i-1}}(P_1) + \left(\frac{c_i}{2}(1-c_1) + \frac{c_i^2}{8}\right) \sum_{k=2}^{5} \left(P_{G_{i-1}}(P_k)\right) + \frac{c_i^2}{4} P_{G_{i-1}}(P_6) \\
P_{G_i}(P_2) &= \left(\frac{c_i}{2}(1-c_1) + \frac{c_i^2}{8}\right) \sum_{k=1,6} \left(P_{G_{i-1}}(P_k)\right) + \frac{c_i^2}{16}\left(3P_{G_{i-1}}(P_3) + 2P_{G_{i-1}}(P_4) + P_{G_{i-1}}(P_5)\right) \\
&\quad + \left((1-c_i)^2 + c_i(1-c_i) + \frac{3c_i^2}{8}\right) P_{G_{i-1}}(P_2) \\
P_{G_i}(P_3) &= \left(\frac{c_i}{2}(1-c_1) + \frac{c_i^2}{8}\right) \sum_{k=1,6} \left(P_{G_{i-1}}(P_k)\right) + \frac{c_i^2}{8} \sum_{k=2,4,5} \left(P_{G_{i-1}}(P_k)\right) \\
&\quad + \left((1-c_i)^2 + c_i(1-c_i) + \frac{3c_i^2}{8}\right) P_{G_{i-1}}(P_3) \\
P_{G_i}(P_4) &= \left(\frac{c_i}{2}(1-c_1) + \frac{c_i^2}{8}\right) \sum_{k=1,6} \left(P_{G_{i-1}}(P_k)\right) + \frac{c_i^2}{16}\left(2P_{G_{i-1}}(P_2) + P_{G_{i-1}}(P_3) + 3P_{G_{i-1}}(P_5)\right) \\
&\quad + \left((1-c_i)^2 + c_i(1-c_i) + \frac{3c_i^2}{8}\right) P_{G_{i-1}}(P_4) \\
P_{G_i}(P_5) &= \left(\frac{c_i}{2}(1-c_1) + \frac{c_i^2}{8}\right) \sum_{k=1,6} \left(P_{G_{i-1}}(P_k)\right) + \frac{c_i^2}{8} \sum_{k=2,3,4} \left(P_{G_{i-1}}(P_k)\right) \\
&\quad + \left((1-c_i)^2 + c_i(1-c_i) + \frac{3c_i^2}{8}\right) P_{G_{i-1}}(P_5) \\
P_{G_i}(P_6) &= \left((1-c_i)^2 + \frac{c_i^2}{4}\right) P_{G_{i-1}}(P_6) + \left(\frac{c_i}{2}(1-c_1) + \frac{c_i^2}{8}\right) \sum_{k=2}^{5} P_{G_{i-1}}(P_k) + \frac{c_i^2}{4} P_{G_{i-1}}(P_1)
\end{aligned}$$

2.14 The Use of Simulated Annealing for Searching for the Best Order.

Finding the order of genetic markers is a combinatorial optimization problem. Simulated annealing has long been used quite successfully for solving combinatorial optimization problems [35]. In this paper, we use simulated annealing to find the best order of genetic markers. Simulated annealing has been used previously [17], to reconstruct chromosomes based on binary scoring of DNA fragments and a Hamming distance-based objective function as well as in genetic mapping [67]. In our case, the objective function is the likelihood of a particular order of genes on a genetic map obtained upon convergence of the EM algorithm. So, in our case a single computation of the objective function for one order of markers is quite expensive. A heuristic stochastic strategy to hunt for a good order is:

1. Generate a random order of probes, Π, and calculate $f(\Pi)$.

2. Select a random segment within the ordering Π.

3. Perform a segment reversal and name the new ordering Π'.

4. Compute $f(\Pi')$.

5. If $f(\Pi')$ is less than $f(\Pi)$, then retain the new order. However, if $f(\Pi')$ is larger than $f(\Pi)$, then generate a random number between 0 and 1. If this random number is less than $E(-(f(\Pi') - f(\Pi))/T$, then retain the new order. Here T denotes the "temperature" of the annealing schedule.

6. Proceed downward in the multiplicative steps, decreasing T by a factor F. Hold each value of T constant until M re-orderings have been attempted or S successful re-orderings have resulted, whichever comes first. If the number of successes equals zero for a given step, the process is complete; otherwise go to step 2.

2.15 Genetic Mapping from the RFLP data of *NEUROSPORA CRASSA*

2.15.1 Calibration of the Annealing Parameters

In order to find the optimal values of the factors T, F, M and S to use the annealing machine for chromosome VII of *Neurospora crassa* we undertake a full factorial design [69] of the four factors (T, F, M, S) for several values for each parameter as shown in the table below. For each combination of the factors the likelihood measure of the converged order was noted in a simulation study. Missing data are handled as described in [63]. We first estimated θ using equation(2.29) on the data (The entire 31 genes in chromosome VII of *Neurospora crassa*) and then used those estimates to simulate a data set according to model equation(2.1) by basically a single inversion when there is a difference (see Figure 2.5 and Figure 2.6). For each combinations of factors in Table 2.7 10 random permutations of the gene order were run, and the likelihood of their converged order was obtained.

There were no significant main effects for any of the factors except for near significance of M (P value < 0.066). All the interactions were insignificant. From the annealing schemes we recommend $T = 10.0$, $F = 0.5$, $M = 100$ and $S = 20$, since this annealing machine

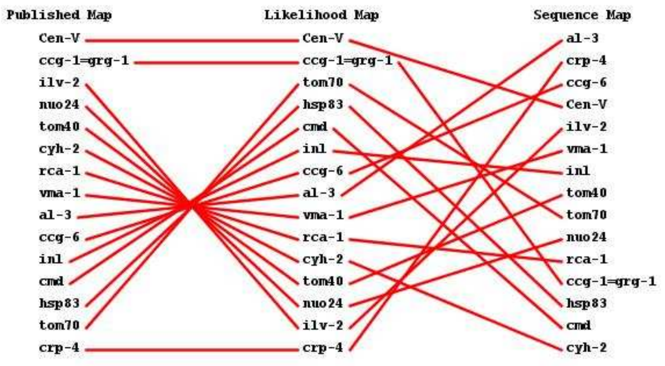


Figure 2.5: Comparative Genetic Map for Linkage Group V

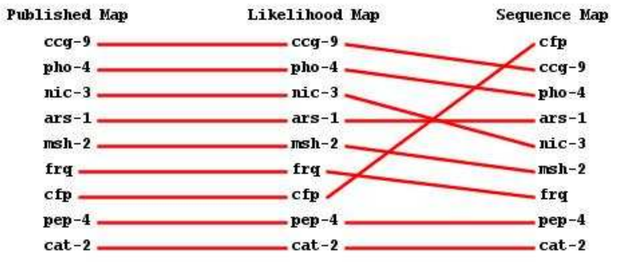


Figure 2.6: Comparative Genetic Maps for Linkage Group VII

converged with the highest likelihood of the converged order and took the least amount of time.

2.15.2 Genetic Map of Linkage Groups V & VII of *Neurospora crassa*

We search for an optimal order of genetic markers of chromosome VII in *Neurospora crassa*([34], [46]) using simulated annealing to maximize the log-likelihood, in Section(2.14). The values of the parameters used in the search of the best order are $T = 10.0$,$F = 0.5$,$M = 100$ and $S = 20$. For each order proposed, the exchange probability estimates are computed by the EM algorithm in Section(2.11.2). The exchange probability estimates with their standard errors and the corresponding recombination fractions (see Section 2.6.3) are reported for the best order in Table 2.10 (See at the end of the paper). We found the covariances of the exchange probabilities small (though not reported here), and that explains why the pairwise analysis provides a good initialization to multiple marker MLE as well as satisfies the independence assumption underlying the goodness-of-fit statistic. For linkage group-VII, the published order [46] turned out to be the best order with likelihood -60.3270 using simulated annealing but the P-value is 0.00026 for the chi-squared goodness-of-fit (See Section2.13). So, the chromosome VII map appears to have a poor fit to the model. For all other linkage groups the P-value indicated a good fit to the model. Goodness-of-fit value for the entire genome (all 7 chromosomes) are tabulated in Table 2.8.The maps for all the chromosomes can be found at *http://gene.genetics.uga.edu*. Linkage groups II, IV and V differed from the published order [46] by a simple inversion as shown for linkage group in Figure(2.5). As can be seen from Table2.9, even in the cases of linkage groups II and V the likelihood of the published order is extremely close to the maximum likelihood of the inferred order. There are at least two reasons for the discrepancy. The simplest is that the maps were constructed by hand, so there is only a limited number of possible solutions considered. Metzenberg(personal communication) has also indicated that he held additional data that helped him decide on the map, data which the current computation did not have access to.

As a final test of the maximum likelihood methodology developed here for building genetic maps, we now demonstrate the methodology generates the correct order of genetic markers on four of the linkage groups. The 277 markers on all linkage groups were mapped onto the genomic sequence of this model system [24]. The markers on this physical map provided by the genomic sequence can then be ordered independently of the genetic map. As can be seen in Figure(2.5,2.6), the two orders of the physical and genetic maps are largely in agreement; see *http://gene.genetics.uga.edu* for figures similar to Figure(2.5,2.6) for four linkage groups. The numbers of crossed lines (discrepancies between genetic and physical maps) for each linkage group are reported in Table2.9 with linkage group V having the most discrepancies between the order of the genetic map and sequence map. The conclusion is that that the genetic map generated by the methodology in this paper is independently corroborated by the genomic sequence except for linkage groups I, II and V. It is not surprising to see the discrepancies for linkage groups I and V, since these are the largest and hence hardest to assemble correctly [34]. The discrepancy in the genomic sequence of linkage group II appears to be due to the assignment of one segment to a different arm of linkage group II. In conclusion, genetic maps with 277 markers can be constructed by the method of maximum likelihood as demonstrated here, and the order inferred by the method of maximum likelihood can be independently corroborated.

2.16 Conclusions and Future Directions

We modeled the recombination process of fungal systems for both random spore and tetrad data. Assuming no-chromatid-interference (NCI) model, a probability model framework was developed using bivalent configurations along the chromosome for 2 genes, and MLE estimators for both random and tetrad data have been studied. It was shown that the MLE estimator of recombination for tetrad data is uniformly more efficient over that from random spore data by usually a factor of at least 4. The probability framework was generalized for an arbitrary number of genes, and the MLE with its standard error have been computed using the EM algorithm. We implemented our model with data on the whole genome of *Neurospora crassa*. Simulated annealing was used to search for the best order of genetic markers and the standard Pearson chi-squared goodness-of-fit values supported the model assumptions (See Table 2.8). A desired extension is to unite this framework with that for constructing a physical map([33], [8])to produce integrated maps. These data are available [34], and the statistical methodology for generating these integrated maps by the method of maximum likelihood are not developed. It would be desirable to develop faster methods [31] for computing the MLE of the genetic map (as in [8], [63] for the physical map and genetic map respectively). It would be useful to have robust alternatives to the MLE [68] as well.

All of these tools are being made available for the first time to the fungal genetics community. The no-chromatid-interference (NCI) is evaluated using this new statistical methodology on the *Neurospora crassa* genome. Under this model the pair of chromatids involved in a double-stranded breaks are equally likely. Based on goodness-of-fit measure, the assumption appears plausible (See Table 2.8). The assumption may need reevaluation as higher density maps come available.

The problem considered here has a broader context. While we have solved the problem for hundreds of markers, what is needed is to solve the problem for thousands of markers to understand the genetic basis of human disease using resources such as the International Hap Map [15]. It will be interesting to see whether or not the methods here scale [62]. The results here suggest the scaling can be done.

2.17 Acknowledgements

The following work is dedicated to David Perkins, who has served as an inspiration to many young scientists. The research was supported in part by a grant from the U.S. Dept. of Agriculture under the NRI Competitive Grants Program (Award No: GEO-2002-03590) to Drs. Bhandarkar and Arnold. We thank the College of Agricultural and Environmental Sciences, University of Georgia for their support.

2.18 Appendix

1. **Proof of Lemma2.2:** Eqn. 2.13 is a cubic polynomial equation and we know that all of its roots are real iff,

$$H < 0 \quad \text{and} \quad G < 0$$

where,

$$H = a_3a_1 - a_2^2 \text{ and } G = a_3^2a_0 - 3a_3a_2a_1 + 2a_2^3$$

Now,

$$\begin{aligned}
a_3 &= -30t_1 - 30t_2 - 30t_3 \\
a_2 &= 64t_1 + 68t_2 + 88t_3 \\
a_1 &= -32t_1 - 56t_2 - 88t_3 \\
a_0 &= 16t_2 + 32t_3
\end{aligned}$$

So,

$$\begin{aligned}
H(\mathbf{t}) &= \frac{1}{9}\left\{-1216t_1^2 - 784t_1t_2 - 464t_1t_3 + 416t_2^2 + 992t_2t_3 + 176t_3^2\right\} \\
&= \frac{1}{9}\left[(t_3 - t_1)(784t_2 + 176t_3) + t_3(208t_2 - 288t_1)\right] \\
&< 0 \quad \text{, Since } t_1 > t_2 > t_3
\end{aligned}$$

Now, $G(\mathbf{t})$ is obtained after some algebraic simplification to be,

$$\begin{aligned}
G(\mathbf{t}) = {} & \frac{113459200}{3}t_1^3t_2t_3^2 - \frac{65945600}{3}t_1t_2^3t_3 - 18022400t_1t_2^4t_3 \\
& -15974400t_1^4t_2t_3 + 45465600t_1t_2t_3^4 + \frac{223232000}{3}t_1^2t_2t_3^3 + \frac{19558400}{3}t_1^2t_2^2t_3^2 \\
& +4300800t_1^3t_2^2t_3 + \frac{56934400}{3}t_1t_2^2t_3^3 - \frac{58777600}{3}t_1^2t_2^3t_3 + \frac{18022400}{3}t_2^4t_3^2 \\
& \frac{31129600}{3}t_2^3t_3^3 + \frac{28672000}{3}t_2t_3^5 - \frac{4096000}{3}t_2^5t_1 + 3379200t_2^4t_1^2 + \frac{6553600}{3}t_2^5t_3 \\
& \frac{45670400}{3}t_2^3t_1^3 + 3379200t_1^4t_2^2 + \frac{9420800}{3}t_1^4t_3^2 + 44646400t_1^3t_3^3 + 48332800t_1^2t_3^4 \\
& -19660800t_1^5t_3 + \frac{63488000}{3}t_1t_3^5 - \frac{45875200}{3}t_1^5t_2 + \frac{36454400}{3}t_2^2t_3^4 \\
& \frac{1638400}{3}t_2^6 + \frac{10240000}{3}t_3^6 - \frac{26214400}{3}t_1^6
\end{aligned}$$

2. **Proof of Eqn.(2.14):** From Eqn. 2.11 we get the log-likelihood function as

$$\text{log}L(c) = t_1\text{log}\left\{4(1-c)^2 + c^2\right\} + t_2\text{log}\left\{c^2 + 4c(1-c)\right\} + t_3\text{log}c^2$$

Now differentiating the above we obtain,

$$\begin{aligned}
\frac{\partial \text{log}L(c)}{\partial c} &= c\left\{\frac{10t_1}{4(1-c)^2 + c^2} - \frac{6t_2}{c^2 + 4c(1-c)}\right\} \\
&\quad + \frac{2t_3}{c} + \frac{4t_2}{c^2 + 4c(1-c)} - \frac{8t_1}{4(1-c)^2 + c^2}
\end{aligned}$$

Hence,

$$\begin{aligned}\sigma^2_{mle.td} &= \frac{(\phi(c)')^2}{-E\left(\frac{\partial^2 \log L(c)}{\partial c^2}\right)}\\ &= \frac{(1-c)^2}{(\sum_{i=1}^{3} t_i)\left[\frac{1}{2} + \frac{50c^2-80c+24}{16(1-c)^2+c^2} + \frac{18c^2-24c+16}{2c^2+8c(1-c)}\right]}\end{aligned}$$

3. **Proof of Eqn.(2.15):** From Eqn. 2.11 we have the probability distribution of T. The heuristic estimator of recombination fraction is given by

$$r_{heu} = \frac{0.5t_2 + t_3}{(\sum_{i=1}^{3} t_i)}$$

Now,

$$\begin{aligned}E\left(r_{heu}\right) &= 0.5P(T_2 = t_2) + P(T_3 = t_3)\\ &= 0.5\left\{\frac{c^2}{2} + 2c(1-c)\right\} + \frac{c^2}{4}\\ &= c\left(1 - \frac{c}{2}\right)\end{aligned}$$

Hence, r_{heu} is an unbiased estimator of r.

$$\begin{aligned}\text{Var}\left(r_{heu}\right) &= \frac{1}{(\sum_{i=1}^{3} t_i)^2}\left[\frac{\text{Var}(T_2)}{4} + \text{Var}(T_3) + \text{Cov}(T_2, T_3)\right]\\ (\sum_{i=1}^{3} t_i)\text{Var}\left(r_{heu}\right) &= \frac{1}{4}\left[\left\{\frac{c^2}{2} + 2c(1-c)\right\}\left\{1 - \frac{c^2}{2} - 2c(1-c)\right\}\right]\\ &\quad + \frac{c^2}{4}\left(1 - \frac{c^2}{4}\right) - \left\{\frac{c^2}{2} + 2c(1-c)\right\}\frac{c^2}{4}\\ &= -\frac{c^4}{4} + c^3 - \frac{9}{8}c^2 + \frac{c}{2}\end{aligned}$$

So,

$$\text{Var}\left(r_{heu}\right) = \frac{-\frac{c^4}{4} + c^3 - \frac{9}{8}c^2 + \frac{c}{2}}{(\sum_{i=1}^{3} t_i)}$$

4. **Proof of Eqn.(2.16):** From Eqn. 2.4 we get the log-likelihood for c in the random spore model as

$$\log L(c) = (n_1 + n_2)\log(2 - 2c + c^2) + (n_3 + n_4)\log(2c - c^2)$$

Differentiating the above we get,

$$\frac{\partial \log L(c)}{\partial c} = (2c - 2)\left[\frac{n_1 + n_2}{2 - 2c + c^2} - \frac{n_3 + n_4}{2c - c^2}\right]$$

Hence the information on c in the model is

$$\begin{aligned} I(c) &= E\left(-\frac{\partial^2 \log L(c)}{\partial c^2}\right) \\ &= \frac{4(\sum_{i=1}^{4} n_i)(1-c)^2}{(2 - 2c + c^2)(2c - c^2)} \end{aligned}$$

Hence,

$$\sigma_{r_{mle}} = \frac{(2 - 2c + c^2)(2c - c^2)}{4(\sum_{i=1}^{4} n_i)}$$

Table 2.5: Possible Tetrad Patterns(taken from Fig 2 in [63])

Source Pattern	**Generated Patterns**	**Sample Points**
P_1	P_1	(0,0),(1,1),(2,2),(3,3),(4,4)
	P_2	(0,1),(1,0),(1,2),(1,4)
	P_3	(0,2),(2,0),(2,1),(2,3)
	P_4	(0,3),(3,0),(3,2),(3,4)
	P_5	(0,4),(4,0),(4,1),(4,3)
	P_6	(1,3),(3,1),(2,4),(4,2)
P_2	P_1	(0,1),(1,0),(2,1),(4,1)
	P_2	(0,0),(0,2),(2,0),(0,4),(4,0),(1,1),(2,2),(3,3),(4,4),(4,2),(2,4)
	P_3	(1,2),(3,4)
	P_4	(1,3),(3,1)
	P_5	(1,4),(3,2)
	P_6	(0,3),(3,0),(2,3),(4,3)
P_3	P_1	(0,2),(2,0),(1,2),(3,2)
	P_2	(2,1),(2,3),(4,3)
	P_3	(0,0),(0,1),(1,0),(0,3),(3,0),(1,1),(2,2),(3,3),(4,4),(1,3),(3,1)
	P_4	(4,1)
	P_5	(4,2),(2,4)
	P_6	(0,4),(4,0),(1,4),(3,4)
P_4	P_1	(0,3),(3,0),(2,3),(4,3)
	P_2	(1,3),(3,1)
	P_3	(1,4),(3,2)
	P_4	(0,0),(0,2),(2,0),(0,4),(4,0),(1,1),(2,2),(3,3),(4,4),(4,2),(2,4)
	P_5	(1,2),(3,4)
	P_6	(0,1),(1,0),(2,1),(4,1)
P_5	P_1	(0,4),(4,0),(1,4),(3,4)
	P_2	(4,1)
	P_3	(4,2),(2,4)
	P_4	(2,1),(2,3),(4,3)
	P_5	(0,0),(0,1),(1,0),(0,3),(3,0),(1,1),(2,2),(3,3),(4,4),(1,3),(3,1)
	P_6	(0,2),(2,0),(1,2),(3,2)
P_6	P_1	(1,3),(3,1),(4,2),(2,4)
	P_2	(0,3),(3,0),(3,2),(3,4)
	P_3	(0,4),(4,0),(4,1),(4,3)
	P_4	(0,1),(1,0),(1,2),(1,4)
	P_5	(0,2),(2,0),(2,3),(2,1)
	P_6	(0,0),(1,1),(2,2),(3,3),(4,4)

Table 2.6: Probability Distributions of the Tetrad Patterns

Source Pattern	**Generated Patterns**	**Sample Points**
P_1	P_1	$(1-c)^2+\frac{c^2}{4}$
	P_2	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
	P_3	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
	P_4	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
	P_5	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
	P_6	$\frac{c^2}{4}$
P_2	P_1	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
	P_2	$(1-c)^2+c(1-c)+\frac{3c^2}{8}$
	P_3	$\frac{c^2}{8}$
	P_4	$\frac{c^2}{8}$
	P_5	$\frac{c^2}{8}$
	P_6	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
P_3	P_1	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
	P_2	$\frac{3c^2}{16}$
	P_3	$(1-c)^2+c(1-c)+\frac{3c^2}{8}$
	P_4	$\frac{c^2}{16}$
	P_5	$\frac{c^2}{8}$
	P_6	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
P_4	P_1	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
	P_2	$\frac{c^2}{8}$
	P_3	$\frac{c^2}{8}$
	P_4	$(1-c)^2+c(1-c)+\frac{3c^2}{8}$
	P_5	$\frac{c^2}{8}$
	P_6	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
P_5	P_1	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
	P_2	$\frac{c^2}{16}$
	P_3	$\frac{c^2}{8}$
	P_4	$\frac{3c^2}{16}$
	P_5	$(1-c)^2+c(1-c)+\frac{3c^2}{8}$
	P_6	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
P_6	P_1	$\frac{c^2}{4}$
	P_2	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
	P_3	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
	P_4	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
	P_5	$\frac{c}{2}(1-c)+\frac{c^2}{8}$
	P_6	$(1-c)^2+\frac{c^2}{4}$

Table 2.7: Levels Examined in the Calibration of the Annealing Machine

Parameter	**Values**
T	10.0 , 20.0
F	0.25 , 0.5 , 0.75
M	100 , 200 , 300
S	0.2M , 0.4M

Table 2.8: Goodness-of-fit for the whole Genome of *N.crassa*

Linkage Group	**Chi-Square**	**df**	**P-Value**
I	71.9640	104	0.9929
II	55.4553	76	0.9633
III	18.5361	60	0.99
IV	57.8726	76	0.9396
V	82.9384	104	0.9364
VI	58.3165	60	0.5374
VII	105.5206	60	0.00026

Table 2.9: Discrepancy and Likelihood Measures for the whole Genome of *N.crassa*

Linkage Group	**Discrepancy**[a]	**Published Map**	**Inferred Map**	**Sequence Map**
I	13	-42.01182	-42.01182	-233.9733
II	13	-63.7872	-63.7825	-62.4309
III	3	-101.1678	-101.1678	-53.4956
IV	9	-93.1492	-86.1172	-143.6690
V	14	-4.7713[b]	-4.7713[c]	-20.7591
VI	7	-73.0293	-73.0293	-79.2024
VII	6	-60.3270	-60.3270	-99.0543

[a]Discrepancy is measured as the total number of line crosses between the Inferred map and the Sequence map.

[b]Actual Value -4.771375255655365

[c]Actual Value -4.771375255655364

Table 2.10: Best Genetic Map with Estimates of Exchange Probability (c), its Standard Error(σ_c) and Recombination Fraction (r) on Linkage Group-VII of *N.crassa*

Genes	c^a	σ_c	r	**Genes**[b]	c	σ_c	r
Tel VIIL–AP8e.1	0.0001	0.0003	0.0001	*AP36a.4–ars-1*	0.0001	0.0042	0.0001
AP8e.1–5:5A	0.0001	0.0003	0.0001	*ars-1–Cen VII*	0.0001	0.0042	0.0001
5:5A–00003	0.4712	0.288	0.3602	*Cen VII–msh-2*	0.16	0.1197	0.1472
00003–ccg-9	0.0001	0.0041	0.0001	*msh-2–frq*	0.0001	0.0043	0.0001
ccg-9–pho-4	0.0001	0.0041	0.0001	*frq–cfp*	0.1568	0.1181	0.1445
pho-4–nic-3	0.0001	0.0041	0.0001	*cfp–AP32a.1*	0.3209	0.237	0.2694
nic-3–AP12i.2	0.0001	0.0041	0.0001	*AP32a.1–R29.1*	0.0001	0.0027	0.0001
AP12i.2–AP11c.3	0.0001	0.0041	0.0001	*R29.1–pep-4*	0.1571	0.2053	0.1448
AP11c.3–AP34a.1	0.0001	0.1154	0.0001	*pep-4–cat-2*	0.3295	0.2812	0.2752
AP34a.1–AP39a.2	0.3128	0.1681	0.2639	*cat-2–COXVIII*	0.3259	0.2032	0.2728
AP39a.2–AP31a.2	0.1553	0.1174	0.1432	*COXVIII–pRB22*	0.0001	0.0037	0.0001
AP31a.2–X23:9G	0.0001	0.0042	0.0001	*pRB22–Ncr-1*	0.0001	0.0037	0.0001
X23:9G–R58.1	0.0001	0.0042	0.0001	*Ncr-1–NP4A9*	0.1568	0.1143	0.1445
R58.1–AP36a.1	0.0001	0.0042	0.0001	*NP4A9–AP5i.2*	0.1483	0.1599	0.1373
AP36a.1–AP36a.4	0.0001	0.0042	0.0001	*AP5i.2–Ncr-9*	0.6519	0.2664	0.4394

[a]c, σ_c and r correspond to the genetic interval formed by the genes at the current row and the following row

[b]comes after the first column for genes is completed

Chapter 3

DESIGN AND ANALYSIS OF AN EFFICIENT RECURSIVE LINKING ALGORITHM FOR CONSTRUCTING LIKELIHOOD BASED GENETIC MAPS FOR A LARGE NUMBER OF MARKERS [1]

[1]Tewari,S., Bhandarkar,S.M. and Arnold, J. 2008
Journal of Bioinformatics and Computational Biology. 5 : 201-250
Reprinted here with permission of publisher

3.1 Abstract

A multi-locus likelihood of a genetic map is computed based on a mathematical model of chromatid exchange in meiosis that accounts for any type of bivalent configuration in a genetic interval in any specified order of genetic markers. The computational problem is to calculate the likelihood (L) and maximize L by choosing an ordering of genetic markers on the map and the recombination distances between markers. This maximum likelihood estimate (MLE) could be found either with a straightforward algorithm or with the proposed recursive linking algorithm that implements the likelihood computation process involving an iterative procedure, called *Expectation Maximization*(EM). The time complexity of the straightforward algorithm is exponential without bound in the number of genetic markers, and implementation of the model with a straightforward algorithm for more than 7 genetic markers is not feasible, thus motivating the critical importance of the proposed recursive linking algorithm. The recursive linking algorithm decomposes the pool of genetic markers into segments and renders the model implementable for hundreds of genetic markers. The recursive algorithm is shown to reduce the order of time complexity from exponential to linear in the number of markers. The improvement in time complexity is shown theoretically by a worst-case analysis of the algorithm and supported by run time results using data on linkage group-II of the fungal genome *Neurospora crassa*.

3.2 Introduction

High density linkage maps are an essential tool for characterizing genes in many systems, fundamental genetic processes, such as genetic exchange between chromosomes, as well as the analysis of traits controlled by more than one gene (*i.e.*, complex traits) [40]. Since genetic maps are most often the critical link between phenotype (what a gene or its product does) and the genetic material, genetic maps can be exploited to address how the genetic material controls a particular trait [21] controlled by one or more genes. Most model systems possess high density linkage maps that can assist in the analysis of complex traits. The bread mold, *Neurospora crassa* [19] , which gave us the biochemical function of genes, is no exception [46]. One approach to understanding the genetic basis of a complex trait is to follow its segregation in offspring along with an array of genetic markers. One class of genetic markers frequently used are restriction fragment length polymorphisms (RFLPs), markers in the DNA itself. Another class are single nucleotide polymorphisms (SNPs). These markers, in essence, allow a triangulation on loci in the DNA affecting the complex trait of interest. Part of this triangulation process involves the construction of the relative positions of these several markers along a genetic map. This is a computationally challenging problem [39] and at the heart of understanding complex traits, such as human disease. In this paper we address the problem of genetic map reconstruction from a large number of RFLP markers. We focus on map construction for a model system *N. crassa* [3], where there is a wealth of published information about how markers segregate because the genetic makeup of gametes (as opposed to offspring) can be identified [3]. In this setting we can build a much more realistic model of the recombination process between chromosomes than in more complex eukaryotes [56].

Given l markers or genetic loci, each with two or more alternate forms of a gene called alleles, the number of distinct types of offspring is 2^l. This implies that the computational complexity of a likelihood-based approach to estimating a genetic map would appear to scale at least as $O(25^{l-1})$ [62]. At first sight the computational complexity of following the segregation of l markers to build a genetic map seems infeasible beyond about 7 markers. It is remarkable that Lander and Green [39] were able to solve this problem for a special case with an algorithm whose computational complexity is linear in l. The likelihood, in their algorithm, is computed from a recombination fraction defined on each genetic interval.The limitation of their work is that they did not model the recombination process in detail; for example, chromatid exchanges are not explicitly modeled as discussed in the next section. Others have tried to circumvent the computational complexity of this problem by utilizing only pairwise information on genetic loci [45]. Solving this reconstruction problem is essential for geneticists to make use of the recently completed International HapMap [15] with thousands of markers scattered throughout the human genome to hunt down important disease causing genes. In addition, several model systems now possess the data for constructing dense genetic maps with thousands of markers [70]. Using such mapping data together with variation in a complex trait, such as heart disease, will allow researchers to triangulate on genes determining the trait of interest. Here we focus on solving this problem in a setting considered ideal for geneticists. For some organisms, it is possible to observe the gametes (as opposed to the offspring) from a parent organized into a package called a tetrad [53] uncomplicated by what is going on in another parent. For organisms such as fungi and some more complex eukaryotes engineered to have this property [18], gametes can be observed directly, as opposed to offspring with the contributions of the two parents. For this reason, we focus on reconstructing a genetic map where tetrads can be obtained. In fungi, such as *Neurospora crassa*, the gametes can be typed directly (Figure 3.1). A string of spores (or gametes) in Figure 3.1 are the products of single cross. In this setting the recombination process can be modeled in detail. The challenge we address here is, in the best of all possible worlds can we reconstruct a genetic map with many markers? This is a very old and difficult problem without a good solution (particularly when the order of markers is unknown) in spite of the fact that hundreds of fungal laboratories around the world make use of this kind of tetrad data in genetic analysis. Zhao and Speed([75], [73]) demonstrate a solution for few number of markers (less than 10) under some assumptions on the exchange process.

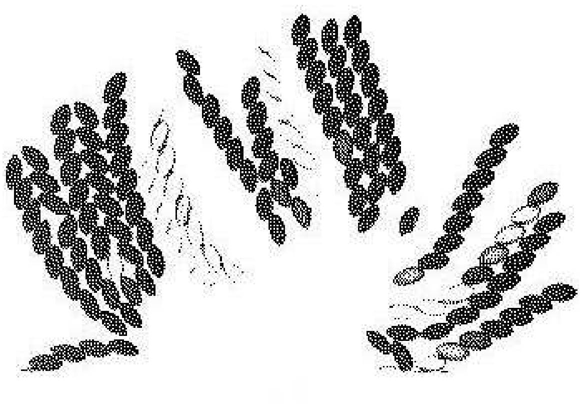


Figure 3.1: Various forms of tetrads : (a) unordered; (b) linear; (c) normally maturing asci of *Neurospora crassa*. (from Namboori B. Raju, [53] and Davis [19])

3.3 Background Material on Meiosis, Recombination and Exchange

The following biological section is introduced to make the paper self-contained. The vast majority of genes in cells with nuclei (eukaryotes) occur on chromosomes. Depending on the organism and life stage, a eukaryotic species can have 1 to several copies of these chromosomes. For example, many eukaryotes, such as animals and flowering plants, possess 2 copies of their chromosomes and are referred to as *diploid*, while many fungi and algae may possess 1 copy of their chromosomes and are referred to as *haploid*. The ploidy for these organisms can vary with their life cycle. For example, diploid animals and plants produce haploid gametes; conversely, haploid fungi often produce a temporary diploid stage (a *meiocyte*) during sexual reproduction. The letter n is used to refer to the number of distinct chromosomes in a haploid condition. So, gametes have n chromosomes, and diploids have $2n$ chromosomes.

The process of recombination underlies the reconstruction of genetic maps and thought to take place principally during a cell division process called *meiosis* underlying sexual reproduction. To understand how recombination takes place between chromosomes, it is necessary to have an understanding of how meiosis proceeds. In eukaryotes, meiosis is the production of specialized cells (such as sperm and eggs) called gametes. Prior to the onset of meiosis the genetic material is completely duplicated, and during meiosis two divisions take place to convert a diploid cell, for example, with $2n$ chromosomes into four haploid gametes called a tetrad, each with n chromosomes. This process is termed meiosis.

As a diploid cell, a meiocyte, enters meiosis, the meiocyte contains at least four copies of each chromosome because DNA replication has taken place already. In the succeeding rounds of two cell divisions, the number of chromosomes in each daughter cell is reduced to one copy. The four copies of each chromosome in the diploid meiocyte are referred to as *chromatids*. During the earliest stage of meiosis, *prophase*, these four chromatids align into a structure called a *bivalent* (See Fig(3.2)). Two pairs of these chromatids are nearly identical, one being descended from the other by DNA replication immediately prior to meiosis. The related chromatids are called *sister chromatids* (same color in Figure 3.2), and if the two chromatids are not related by the immediately prior round of DNA replication, they are *nonsister chromatids*. A bivalent then consists of two pairs of sister chromatids.

The physical proximity of the chromatids in Prophase of meiosis leads to exchanges between chromatids. A exchange can arise as shown in Figure 3.2. These exchanges are generated by random double stranded breaks, in which the ends of chromatids reanneal with the wrong chromatid as shown in Figure 3.2. The resulting exchanges can take place anywhere along the chromatids, and the positions of the exchanges vary from meiocyte to meiocyte (as well as their products, *i.e.*,from gamete to gamete). In that sister chromatids are so similar, exchanges are usually only observed between nonsister chromatids. Let us designate the sister chromatids from one parent as 1 and 2, and the sister chromatids from the other parent, 3 and 4. Exchanges are then usually only observed between 1 and 3, 1 and 4, 2 and 3, and 2 and 4 as depicted in Figure 3.3. It is plausible that exchanges between each of these combinations of nonsister chromatids are equally likely because the exchanges are generated by random breakage events along the chromatids. This hypothesis or assumption is referred to as *No-Chromatid-Interference*(NCI). There is substantial empirical support for this hypothesis [72].

The process of crossing-over underlying meiotic recombination shuffles the allele pairs in the diploid parent and deals them out randomly as the products of meiosis (such as egg and sperm). For clarity, *meiotic recombination* can be defined as the production of gametes from meiosis with genotypes that differ from the parental genotypes that combined to form the (diploid) parental meiocyte. Put more simply, if the descendant does not resemble the parent in its genotype, we say the descendant is a *recombinant*. The product of meiotic recombination is referred to as a recombinant, and the physical process generating a recombinant is hypothesized to be *crossing-over*. Thus, the hypothesis is that the breakage-and-rejoining process leading to exchanges is an explanation for recombination of genes on chromatids, making children different from their parents.

There are two different flavors of meiotic recombination: *independent assortment* of genes on different chromosomes and *crossing-over* between genes on the same chromosome. To see genetic recombination, it is necessary that the pair of genes involved in recombination in the diploid meiocyte each have two different alleles. That is, each gene has to be *heterozygous* in the diploid parental meiocyte. For constructing a genetic map, we only consider genes on the same chromosome, and thus do not consider independent assortment. In Figure 3.2 we present how recombination works. In Figure 3.2, one parental genotype (a.b) is labeled with all mutant alleles, and the other parent is labeled with what is called wild type alleles (A.B). A typical meiotic product from a single exchange (SCO) a.B would be termed a recombinant by the definition above, as the recombinant is genetically distinct from the haploid parents, a.b and A.B. In Figure 3.2(Redrawn from http://www.ncbi.nlm.nih.gov/books/bv.fcgi?rid=iga.figgrp.1135), other double exchange events, such as double exchanges (DCO), lead to different allelic combinations that are recombinant as well. The tetrads from one meiosis can then be classified as *parental ditype* (PD), in which no recombinants occur; as *tetrad type* (T), in which equal numbers of recombinants and non-recombinants co-occur; or as *non-parental ditype* (NPD), in which only recombinants occur.

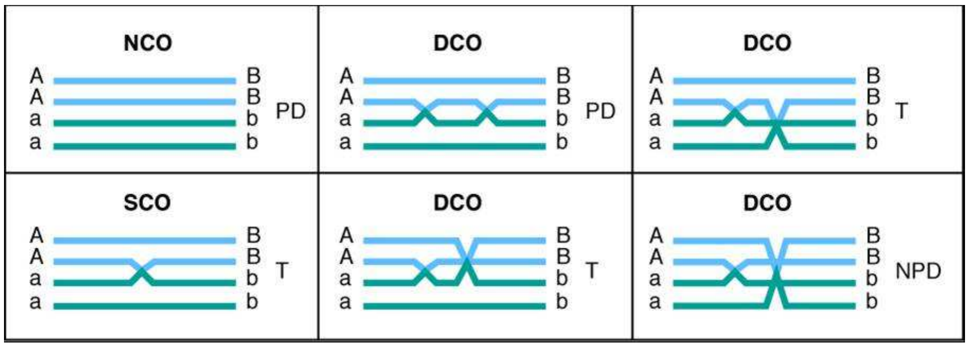


Figure 3.2: The ascus classes produced by exchanges between linked loci. NCO, nonexchange meioses; SCO, single-exchange meioses; DCO, double-exchange meioses. Chromosomes of the same color are sister chromatids

3.4 Model Assumptions in Comparison to Existing Models

One critical feature of the exchanges is how they occur along a chromosome. One assumption that has been made by Lander and Green [39] is that crossovers along a chromosome are independent and uniformly distributed. This assumption of no crossover interference, referred to as the (NChI) hypothesis enabled the likelihood calculation to be tractable in their work [39]. For phase-known type data Lander and Green compute the likelihood of an order of markers by simply computing the fraction of recombinants for each interval. Although this give a likelihood against which to compare marker orders relative to each other, it does not go into the recombination process in detail. They do not consider the chromatid exchanges at all which is crucial to describing recombination. The problem with not having a model is that likelihood measures could differ arbitrarily for small changes in the marker order. The absence of a model also implies that there are no parameters one can monitor in order to examine the changes in the process underlying the model in response to the changes in the marker order.

There have been several approaches to modeling the crossover process in a series of papers by Zhao and Speed (see [41], [73], [75]). For example, they formulate a more general model of crossing-over based on a stationary renewal process [74] to derive the likelihood formulation based on the popular Chi-Square model [22], as a special case. They also perform statistical tests to evaluate the performance of the model for several important data sets [73]. The Chi-square model is obseerved to fit the data well while accounting for interference. However there are a few points that limit their model for broader application. In all of the models that have been proposed so far, the crossover process has been assumed to follow some mathematically tractable process, and several assumptions on the process are generally made to make it viable. For example, in the Chi-Square model [22] the so-called "recombinational intermediates" (C) have been assumed to be uniform across the chromosome (*i.e.*, with no interference) while their resolutions are assumed to follow a particular pattern. Though the model gives a measure of interference through its parameter m, the meaning of the model itself is quite abstract. For example, the model states that a non-exchange resolution(C_0) must result m times after each crossover resolution(C_x) followed by a crossover resolution(C_x).

Zhao and Speed ave worked out very general expressions for the joint probability of multilocus recombinants in Theorem 2.2 of [74], but it is not clear how those can be actually computed except when the assumed process (for example, the Chi-square model) allows sums over all possible exchanges to be represented as explicit closed-form expressions (Appendix: Theorems 1 and 2 in [73]). Even with assumptions that make it mathematically tractable as in [73], there is no analysis presented for more than 10 markers. It is not clear if their method of likelihood maximization (downhill simplex method) would scale for hundreds of markers without running into some kind of computational bottleneck. Though these models are quite helpful in fitting data, we feel that the assumptions underlying their model of the crossover process are hard to verify as exchanges can not be observed directly enough to provide any direct validation. Note that a crossover process is used to connect the observed gametes to the bivalent configurations. Nonetheless, despite their aforementioned limitations, the methods presented by Zhao and Speed do provide insights into recombination.

In this paper we do not make any assumptions regarding the underlying crossover process, if any, that gives rise to the observed gametes. We consider a probabilistic framework for all possible bivalent configurations along the chromosome using the *No Chromatid Interference* (NCI) model, which assumes that chromatid exchange between non-sister chromatids is equally likely. The term *bivalent configuration* describes how chromatids are joined with their non-sister chromatids along the chromosome. Figure 3.2 shows depicts possible bivalent configurations in a tetrad for two intervals. The following sections detail the proposed model formulation and show how connections to key concepts such as, recombination fraction, and likelihood based marker order, can be made using the proposed model.

3.5 A Mathematical Formulation for Bivalent Configurations.

The primary emphasis of this paper is to design and analyze the proposed recursive linking algorithm. In order to make the algorithm more readable, the theory underlying the algorithm, which is under submission as a separate work, is presented in a concise manner. For detailed proofs of the theorems, interested readers are referred to [61].

Before describing the mathematical model in detail, we introduce the following glossary of mathematical terms and symbols used:

- c_i = probability of a non-sister chromatid exchange for the i^{th} genetic interval
- S_i = sample space for the i^{th} genetic interval
- S^l = sample space for all the genetic intervals

- $\phi_k = k^{th}$ unique chromatid exchange in S^l
- $f_k = k^{th}$ genotype
- $n_j =$ cell frequency for the j^{th} observed genotype
- $R_k =$ tetrad obtained via Eqn. 3.5 for exchange ϕ_k
- $p_j =$ cell probability for the j^{th} observed genotype
- $N = d_g =$ total number of distinct genotypes (or cells).
- $n = \sum_{j=1}^{d_g} n_j =$ total number of genotypes observed.

We model the genotypes 1 1 , 0 0,1 0 and 0 1 (where 1 and 0 refer to paternal and maternal alleles respectively) of a genetic interval (say,S_i), as a consequence of two simultaneous independent and identical chromatid exchange events (say, S). Note that this double chromatid exchange is modeled as a discrete event as opposed to an analogous continuous process. The four possible chromatid exchanges between non-sister chromatids are pictorially represented in Figure 3.3. There are 5 possible ways of performing a chromatid exchange (including a non-exchange as one of these possibilities) between non-sister chromatids. Note that our model accounts for all possible bivalent configurations (a bivalent configuration is defined as the chromatid exchanges on four strands of a tetrad) that may result due to any number of exchanges that may occur along the chromosome. It is important to distinguish that our model does not represent crossovers per se, but rather a pair of abstract discrete events that is capable of mapping the bivalent configurations, which could arise from any number of physical exchanges (chiasma). Let c_i denote the probability of a chromatid exchange in S between any two non-sister chromatids in the i^{th} genetic interval S_i at meiosis. Since under the *No-Chromatid-Interference* (NCI) assumption, all chromatid exchanges are equally likely, we get:

$$\begin{aligned} S &= \{0,1,2,3,4\} \\ P(i) &= \frac{c}{4}; i = 1, \cdots, 4; i \in S \\ P(0) &= 1 - c; 0 \in S \end{aligned} \tag{3.1}$$

The element 0 in S indicates the absence of an exchange event. Elements 1, 2, 3 and 4 indicate that non-sister chromatids $(1,3), (2,3), (2,4)$ and $(4,1)$ took part in the exchange process respectively. The set S_i, defined by the Cartesian product $S \times S$, enumerates all possible bivalent configurations for the i^{th} genetic interval.

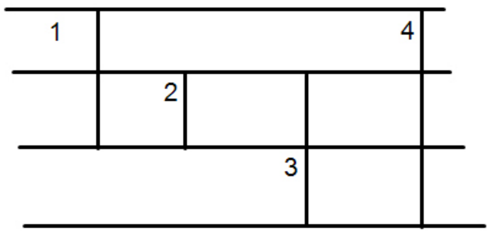


Figure 3.3: The single exchange events are signified by a vertical line, and the exchanges take place between chromatids at the ends of the vertical lines. These 4 strands are found in Prophase-I [19]. All exchanges are equally likely under our model.

3.6 Multi locus Genetic Likelihood for a Specified Order of Genetic Markers.

The probability distribution of S_i denoting bivalent configurations between locus $A_i$and $A_{(i+1)}(i = 1, ..., l-1)$, where l=total number of loci being studied, can be derived as follows using Eqn. 3.1.

$$P(\{i,j\}) = \frac{c_i^2}{16} I_{\{i \neq 0; j \neq 0\}} + \frac{c_i(1-c_i)}{4}\{I_{\{i=0; j\neq 0\}} + I_{\{i \neq 0; j=0\}}\} + (1-c_i)^2 I_{\{i=j=0\}} \tag{3.2}$$

where, $\{i,j\} \in S_i$.

Let ϕ_k denote a unique chromatid exchange on S^l as described below:

$$\phi_k = i_1 \times i_2 \times ... \times i_{l-1} \tag{3.3}$$

where,

$$k = i_1.i_2.i_3...i_{l-1} \ ; \ i_j \in S_i \ ; \ \phi_k \in S^l = \prod_{i=1}^{l-1} S_i$$

From this point on, for the sake of brevity, we may abbreviate term *chromatid exchanges* to simply *exchanges* when referring to ϕ_k.

Let f_k denote a multi-locus genotype with l loci:

$$f_k = i_1 \times i_2 \times ... \times i_{l-1} \times i_l$$

where,

$$k = i_1.i_2.i_3...i_l \quad ; \quad i_j = 0, 1; \quad \forall j = 1, \cdots, l$$

The indices $i_j = 1$ and $i_j = 0$ indicate the paternal and maternal alleles respectively. The progeny are obtained by exchanges between homogeneous parents. The observed data set can be represented as:

$$\mathcal{D} = \left\{ n_j \quad ; \quad \forall j = 1, \cdots, 2^l \right\}$$

where, n_j is the observed frequency of f_j.

3.6.1 Recombination fraction r_i:

There are a variety of ways that various researchers have connected the recombination process to the underlying exchange process. For example, Mather [42] connects the recombination fraction to the number of exchanges occurring on the chromosome. Others, such as Haldane [28], relate the recombination fraction to a physical distance. More recently Zhao *et.al* [73] relate recombination to an underlying exchange process. In the proposed model, the probability of recombination (or recombination fraction) can be calculated simply from the probability distribution of the bivalent configurations in set S_i.

The recombination fraction r_i is defined as follows:

$$\begin{aligned} r_i &= P(\text{at least one meiotic product is recombinant in } S_i) \\ &= c_i(1 - \frac{c_i}{2}) \\ &= \phi(c_i) \end{aligned} \tag{3.4}$$

Also,

$$\underset{c_i\in[0,1]}{\text{Max}}\, r = \underset{c_i\in[0,1]}{\text{Max}}\, \phi(c_i) = \frac{1}{2}, \quad \forall i = 1, \cdots, l-1,$$

which is the theoretical maximum of the recombination fraction from the theory of Mendelian genetics.

3.6.2 Probability distribution on S^l

Let us define the following functions

$$\begin{aligned} f^0(a) &= (a_1, a_2, a_3, a_4)' \\ f^1(a) &= (a_3, a_2, a_1, a_4)' \\ f^2(a) &= (a_1, a_3, a_2, a_4)' \\ f^3(a) &= (a_1, a_4, a_3, a_2)' \\ f^4(a) &= (a_4, a_2, a_3, a_1)' \\ \text{where,} & \\ a &= (a_1, a_2, a_3, a_4)' \\ a_i &= 0, 1 \ \forall i \end{aligned} \tag{3.5}$$

Note that Eqn. 3.5 encodes the elements of set S (See Eqn. 3.1) as mathematical functions. For example, the first function $f^0(a)$ encodes element 0, showing no chromatid exchange whereas function $f^4(a)$ encodes element 4, indicating that the first strand and the fourth strand have had an exchange.

The function $f_{ij}(a) = f_j(f_i(a))$ corresponds to the events in S_i accounting for all possible bivalent configurations. For a particular chromatid exchange ϕ_k we can generate a model tetrad at meiosis using the function f_{ij}. The matrix R_k of size $4 \times l$ defines the simulated tetrad as follows [57]:

$$R_k = \left(R_0 R_1 \cdots R_{(l-1)}\right) \tag{3.6}$$

where,

$$R_0 = (1100)' \quad ; \quad R_i = f_{jk}(R_{i-1}) \, \forall i = 1, \cdots, l-1$$

and the i^{th} genetic interval S_i contains the observed chromatid exchange $\{j,k\}$. In other words, R_k has 4 rows which correspond to the 4 gametes in a tetrad during meiosis if the chromatid exchange ϕ_k had occurred according to our model.

The conditional distribution of f_i for a given ϕ_k is

$$P(f_i|\phi_k) = \frac{1}{4}\sum_{j=1}^{4} I_{f_i \in R_{k(j,.)}} \tag{3.7}$$

where, $R_k(j,.)$is the j^{th} row of R_k.

The marginal density of a single spore f_i is given by

$$\begin{aligned} P(f_i) &= \sum_k P(f_i|\phi_k) \times P_k \\ &= C \times P \end{aligned} \tag{3.8}$$

where, C is the conditional probability matrix given by :

$$\left.\begin{aligned} C &= ((c_{ki})) \\ c_{ki} &= P(f_i|\phi_k) \quad \text{(from equation (3.7))} \end{aligned}\right\} \tag{3.9}$$

and P is given by,

$$\begin{aligned} P &= (P_k;\ \ \forall k)' \\ P_k &= P(\phi_k) \\ &= \prod_{j=1}^{l-1} P\left(I_j = i_{k,j}\right) \end{aligned} \tag{3.10}$$

where, $i_{k,j} \in S_j$ and the probability distribution $P\left(I_j = i_{k,j}\right)$ is as defined in equation (3.2).

Let $\Theta = (c_1, c_2, ..., c_{l-1})'$ denote the unknown parameter vector in the model. The log-likelihood of $\mathbf{f} = (f_i, i = 1, \cdots, 2^{l-1})'$, viewed as a function of Θ, is given by :

$$l(\Theta|D) = \sum_{i=1}^{N} n_i \log\left[\sum_k \left[\frac{1}{4}\sum_{j=1}^{4} I_{\{f_i \in R_k(j,.)\}} \prod_{j=1}^{l-1} P_j\left(I_{k,j} = i_{k,j}\right)\right]\right] \tag{3.11}$$

Note that the log-likelihood in Eqn. 3.11 is distinct from the one in Zhao *et.al* [73]. Zhao and Speed have formulated a log-likelihood function similar to the one in Eqn. 3.11 for ordered tetrads in the appendix of [75]. However, the log-likelihood in Eqn. 3.11 does not hypothesize an exchange process. Both, Lander and Green [39] and Zhao and Speed [75] have used the EM algorithm in the context of genetic map reconstruction.

The following two theorems maximize the log-likelihood in equation (3.11) using a set of recurrence relations obtained via the Expectation-Maximization (EM) algorithm [20]. The proofs are not given in the interest of brevity, but the development of Eqn. 3.12 is described elsewhere [61].

Theorem 3.1. *The EM-iterative equations are given below.*

$$\begin{aligned} \Theta^{(h+1)} &= \left(c_m^{(h+1)} \ \forall m = 1, \cdots, l-1\right)' \\ \textit{where } c_m^{(h+1)} &= \left(\frac{2N_{2,m} + N_{1,m}}{2N_m}\right)^{(h)} \end{aligned} \tag{3.12}$$

where

$$N_m = \sum_k n_k^{(h)} = N_{0,m} + N_{1,m} + N_{2,m}$$

$$N_{0,m} = \sum_{k \,\middle|\, i_{k,m}=(0,0)} n_k^{(h)}$$

$$N_{1,m} = \sum_{\substack{k \,\middle|\, i_{k,m}=(i_1,i_2) \\ i_1=0 \text{ (Strict)OR } i_2=0}} n_k^{(h)}$$

$$N_{2,m} = \sum_{\substack{k \,\middle|\, i_{k,m}=(i_1,i_2) \\ i_1\neq 0 \text{ AND } i_2\neq 0}} n_k^{(h)}$$

$$n_k^{(h)} = \sum_j n_j \pi_{k|j}(\Theta^{(h)})$$

$$\pi_{k|j}^{(h)} = P(x_{kj} = 1|f_j) = \frac{\pi_{j|k}^{(h)} \times \pi_k^{(h)}}{p_j^{(h)}}$$

$$p_j^{(h)} = \sum_k \pi_{j|k}^{(h)} \times \pi_k^{(h)}$$

$$\pi_{j|k}^{(h)} = c_{ki} \textit{ in Eqn. 3.9 for the } h^{th} \textit{iteration}$$

$$\pi_k^{(h)} = P_k \textit{ in Eqn. 3.10 for the } h^{th} \textit{iteration}$$

Note that, $i_{k,m}$ denotes an event in S_m for the exchange ϕ_k.

The following theorem provides an initialization of c for the recurrence relations in Eqn. 3.12.

Theorem 3.2. *Let* $\mathbf{f} = (f_1, f_2, f_3, f_4)'$ *be the observed frequency vector corresponding to all possible meiotic products for parental genes M and O for two markers. The genotype vector for* $\mathbf{f}$ *is (MM MO OM OO)'. The maximum likelihood estimator [54] of the exchange probability c under the model represented by Eqn. 3.1 is unique and is given as follows:*

1. *If* $f_1 + f_4 < f_2 + f_3$ *then* $c_{mle} = 1$
2. *If* $f_1 + f_4 \geq f_2 + f_3$ *then* c_{mle} *is given by the unique solution (in the interval* $[0, 1]$ *) of the following equation:*

$$f(c) = c^2 - 2c + D = 0 \tag{3.13}$$

where,

$$D = \frac{2(f_2 + f_3)}{N} \; ; \; N = \sum_{i=1}^{4} f_i$$

This theorem is used to obtain the starting values of c_m *for the EM-iterative equations in Theorem (3.1).*

3.7 The Straightforward Algorithm

3.7.1 The RFLP Data

Some of the RFLP data for chromosome-I of *Neurospora crassa* [46] are shown in Table 3.1. In the data, at each locus, the symbols ”M” and ”O” denote genes of the parents which have been encoded in the pseudocode description of the algorithms to follow as 1 and 0 respectively. A dash (-) indicates that the scoring was not done or was equivocal and thus denotes a missing observation. Note that in the pseudocode description of the algorithms to follow, any value other than a 0 or 1 indicates a missing value.

Table 3.1: A Partial View of the RFLP Data

	A 1	A 4	B 6	B 7	C 1	C 4	D 5	D 7
AP10.1,AP10.4	M	O	M	O	M	O	M	O
AP8d.4	M	-	M	O	M	O	M	O
AP38a.1,R237	O	M	M	O	M	O	O	O

3.7.2 Treating the Missing Scores

For a particular order of genetic markers (for example, the rows of the Table 3.1) the spores (for example, the columns of the Table 3.1) are searched to see if they contain any missing score. The spores with missing scores are substituted in the following manner. If a missing genotype at any particular locus is surrounded by the same type (for example, genotypes of the pattern M - M or O - O, the missing genotype is substituted by the surrounding genotype. On the other hand, if the missing genotype is surrounded by different genotypes (for example, genotypes of the pattern M - O or O - M, a genotype between O and M is chosen with equal probability and substituted for the missing value. We do not attempt to address the issue of missing observations in this paper beyond this simple adjustment and spores with any other pattern of missing values (such as successive missing genotypes, though very rare) are simply removed from the study. The pseudocode below describes the algorithm used to achieve this. We have used Java syntax [32] throughout this paper in illustrating the various algorithms.

Algorithm : Treating Missing Values

```
temp1=new int[sporeNumber];
for int index=0;index<sporeNumber;index++ do
  count1+=checkMissing(getCol(data,index+1),order,missingValue);
  if checkMissing(getCol(data,index+1),order,missingValue) == 1 then
    temp1[index]=0; // missing Value is present
  else
```

```
      temp1[index]=1;
    end if
  end for
  {Treating for Possible Missing values }
  for int index2=0;index2<sporeNumber;index2++ do
    for int index1=0;index1<geneNumber-1;index1++ do
      if !(data[index1][index2]==0 OR data[index1][index2]==1) AND index1>0 then
        if data[index1-1][index2]==0 AND data[index1+1][index2]==0 then
          data[index1][index2]=0;
        end if
        if data[index1-1][index2]==1 AND data[index1+1][index2]==1 then
          data[index1][index2]=1;
        end if
        if (data[index1-1][index2]==1 AND data[index1+1][index2]==0)
              OR (data[index1-1][index2]==0 AND data[index1+1][index2]==1) then
          if Math.random()<0.5 then
            data[index1][index2]=0;
          else
            data[index1][index2]=1;
          end if
        end if
      end if
    end for
  end for
  {Spores with Persistent Missing values are Removed:}
  count2=0;
  int[][] trimdData1=new int[loci][sporeNumber-count1];
  for int index1=0;index1<sporeNumber;index1++ do
    if temp1[index1]==1 then
    for int index2=0;index2<loci;index2++ do
      trimd_data1[index2][count2]=data[order[index2]-1][index1]
    end for
    count2++;
  end if
end for
{trimdData1 contains usable genotypes for the order specified}
```

3.7.3 Identifying the Distinct Genotypes and Computing their Count

For l markers there are 2^l possible genotypes and as many categories for a multinomial distribution [47] if each genotype is considered a category. Obviously, the categories are mutually exclusive and exhaustive, and the trials are independent if the spores are assumed to have come from different diploid parents. We identify the distinct genotypes and count their number using the following algorithm.

Algorithm : Counting Distinct Genotypes

```
int[][] trimdData2;
int[] freq;
searchList=new int[trimdData1[0].length];
exhaustList=new int[trimdData1[0].length];
for int index=0;index<trimdData1[0].length;index++ do
  searchList[index]=1
  exhaustList[index]=1
end for
compareTo=1;
count3=0;
count4=0;
while counter5==1 do
  for int index1=0;index1<loci;index1++ do
    for int index2=0;index2<searchList.length;index2++ do
```

```
      if searchList[index2]==1 then
        if trimdData1[index1][compareTo-1]!= trimdData1[index1][index2] then
          searchList[index2]=0;
        end if
      end if
    end for
  end for
  for int index3=0;index3<searchList.length;index3++ do
    if searchList[index3]==1 then
      exhaustList[index3]=0;
      count3++;
    end if
    searchList[index3]=exhaustList[index3];
  end for
  count4++; //This is counting the distinct loci patterns
  if count4==1 then
    trimdData2=addCol(getCol(trimdData1,compareTo));
    {addCol(.) returns a column in the type of $trimdData2$}
    freq=addElement(count3);
    {addElement(.) returns $count3$ as a vector (in the type of $freq$)}
  else
    trimdData2=addCol(trimdData2,getCol(trimdData1,compareTo));
    {$trimdData2$ is updated with an additional column}
    freq=addElement(freq,count3);
    {$freq$ is updated with an additional element $count3$}
  end if
  count3=0;
  check=0;
  counter5=0;
```

```
 **for**  check=0;check<searchList.length;check++ **do**
  **if** searchList[check]==1 **then**
   counter5=1;
   break;
  **end if**
 **end for**
 compareTo=check+1; // *compareTo* denotes position rather than array index
**end while**

{*trimdData2* **contains the distinct genotypes in the order of the markers** }
{*freq* **contains the frequency count of the distinct genotypes in** *trimdData2*}
```

3.7.4 Initial Probability Estimates

Theorem (3.2) is used to compute the initial estimates of the exchange probabilities for each interval in the given order of markers. These probability estimates are interval based and do not take into account the interdependence of the markers in the order, but rather serve as good initial estimates to implement the EM algorithm detailed in Section 3.1. The algorithm for computing the initial estimates of c is given below.

```
**Algorithm : Computing Initial Probability Estimates of** $c$
**for** int $i$=0;$i$<loci-1;$i$++ **do**
 int count=0;
 double d,c;
 **for** int index=0;index<sporeNumber;index++ **do**
  **if** (trimdData1[i][index]==1 AND trimdData1[i+1][index]==0)
  OR trimdData1[i][index]==0 AND trimdData1[i+1][index]==1) **then**
   count++;
  **end if**
 **end for**
 d=(double)2*count/obs;
```

```
    **if** d > 1.0 **then**
      cProbOld[index]=1.0;
    **else**
      cProbOld[index]=1-sqrt(1-d);
    **end if**
  **end for**
  {*cProbOld* **holds the initial probability estimates**}
```

3.7.5 Computing the Likelihood and Implementing the EM Algorithm

To compute the likelihood described by Eqn. 3.11, it is evident that we need to process 25^{l-1} exchanges. The algorithm has to be extremely efficient to handle a large number of genetic markers to allow for several computations of the log-likelihood for different orderings of the markers (see Section 3.12).

Given the huge computational complexity of this task, it would not be possible to store either the matrices C or R on account of their prohibitive size. Note that we must compute the likelihood incrementally (and hence on the fly), and the nested FOR loops in algorithm *Loop(0)* must be dynamically created (as l is a variable). Hence a recursive function needs to be designed to accomplish this task. Furthermore, to implement the EM algorithm to iterate the value of the probability vector c described in the series of equations (3.12) one must again process 25^{l-1} exchanges. Consider the following two equations from the series of equations (3.12).

$$\pi_{k|j}^{(h)} = P(x_{kj} = 1|f_j) = \frac{\pi_{j|k}^{(h)} \times \pi_k^{(h)}}{p_j^{(h)}}$$

$$p_j^{(h)} = \sum_k \pi_{j|k}^{(h)} \times \pi_k^{(h)}$$

The computation of $\pi_{k|j}^{(h)}$ requires the knowledge of $p_j^{(h)}$- whose value is based on the computation of 25^{l-1} exchanges. This makes the simultaneous implementation of the computation of the likelihood and running the EM algorithm (to update the exchange probability vector c) a non-trivial problem. This problem has been bypassed by implementing both, the likelihood computation and the EM algorithm in a recursive loop, which avoids computing the cell probabilities $p_j^{(h)}$ until all the exchanges have been processed and readjusts them in the end as they are computed during the likelihood computation. The following algorithm illustrates this point.

Algorithm : Computing the Likelihood and Implementing the EM Algorithm

```
for iteration=0;iteration<iterationLimit;iteration++ do
  probFOld=new double[freq.length];
  cProbNew=new double[loci-1];
  counter=new int[loci-2];
  pCount=new double[3][loci-1][freq.length];
  Loop(0);
  Loop(0) creates both pCount(ie cProbNew) and probFOld at the same time
  /* Reporting the likelihood */
  logLikelihood=0.0;
  for int index=0;index<freq.length;index++ do
    logLikelihood+=freq[index]*Math.log(probFOld[index]);
  end for
  System.out.println(”The like lihood is ”+ logLikelihood +” at iteration ” + iteration);
  /* Getting the posterior counts */
  for int index1=0;index1<3;index1++ do
    for int index2=0;index2<loci-1;index2++ do
      for int index3=0;index3<freq.length;index3++ do
        pCount[index1][index2][index3]=pCount[index1][index2][index3]/probFOld[index3]
      end for
      postCount[index1][index2]=Sum(pCount[index1][index2])
    end for
  end for
  /* Getting the c probs */
  for int index1=0;index1<loci-1;index1++ do
    cProbNew[index1]=$\frac{(\text{postCount[1]}[index1]+2*\text{postCount[2][index1]})}{(2.0*\text{Sum(getCol(postCount,index1+1))})}$
```

```
 end for
 if Convergence(cProbOld,cProbNew,0.0000001) ==1 then
  break;
  {Convergence(.) checks if the absolute difference in all dimensions is < 0.0000001}
 end if
 cProbOld=setEqual(cProbNew);
 {setEqual(.) creates a copy of the object}
end for
```

Algorithm : Loop(0)

```
void Loop2(int index )
if index < loci-2 then
 for counter[index]=0;counter[index]<24;counter[index]++ do
  Loop2(index+1);
 end for
end if
if index == loci-2 then
 for int check1=0;check1<24;check1++ do
  for int check2=0;check2<loci-2;check2++ do
   k[check2]=counter[check2];
  end for
  k[loci-2]=check1;
  r=getRMatrix(k);
  if kIsWorthy(r,trimdData2)==1 then
   prob=kProb(cProbOld,k);
   double[] sum=new double[freq.length];
   for int index1=0;index1<freq.length;index1++ do
    sum[index1]=countMatch(r,getCol(trimdData2,index1+1))
    *prob*freq[index1]/(4.0*totalObs)
```

```
        {$countMatch(.)$ counts the number of tetrads (out of 4) in $r$ that match with the
        current distinct genotype}
        probFOld[index1]+=countMatch(r,getCol(trimdData2,index1+1))/4.0*prob
       **end for**
       **for** int index2=0;index2<loci-1;index2++ **do**
        pCount[CellSpecial(k[index2])][index2]+=sum;
        { The above addition is a vector addition}
       **end for**
      **end if**
    **end for**
  **end if**
```

In the pseudocode above, k denotes a particular exchange ϕ_k as defined in Eqn. 3.3. The function *getRMatrix* implements Eqn. 3.6 to create the R_k matrix corresponding to the exchange ϕ_k. The function $kProb$ computes the marginal probability P_k due to exchange ϕ_k as defined in Eqn. 3.10. The matrix R_k and the probability P_k in turn create matrices C and P progressively during the course of the recursive loop to compute the marginal probability of each observed distinct genotype as defined in Eqn. 3.8. These marginal probabilities along with the counts for the distinct genotypes are used to compute the log-likelihood in Eqn. 3.11. A particular exchange ϕ_k does not enter into the computation of the likelihood so long as it does not have positive probability for at least one distinct genotype. The elimination of such exchanges is achieved with the function $kIsWorthy$, which implies that at least some amount of computation cannot be avoided for each exchange. In the recursive algorithm that we propose in the paper, this feature is handled more efficiently where a large number of exchanges are eliminated by performing checks on a few. In the pseudocode the vector *sum* computes the conditional probabilities across all distinct genotypes. The conditional probability is computed with the help of the function *countMatch* that implements Eqn. 3.7, by counting the number of strands (out of 4) in R_k that match with the observed genotype. The vector $freq$ has the observed count corresponding to the distinct genotypes(d_g), *totalObs* is the total sample size, and *probFOld* stores the marginal probability of each distinct genotype using Eqn. 3.8.The array *pCount* implements the EM algorithm via Eqn. 3.12 by re-categorizing the vector *sum* based on the exchange values along the chromosome. Note that the denominator of $\pi_{k|j}$, i.e., p_j, the marginal probability due to the j^{th} distinct genotype, is omitted from its $\pi_{k|j}$ computation as that requires going through all the exchanges, and is currently being progressively computed by *probFOld*. To compute $n_k^{(h)}$ in Eqn. 3.12 we need to sum up the inverse probabilities $\pi_{k|j}$ across the distinct genotypes but, as their marginal probabilities are not computed yet, it is not possible to do so. We work around this problem by adding another dimension along the number of distinct genotypes to the structure *pCount*. The first dimension of *pCount* is of magnitude 3 to account for $N_{0,m}$,$N_{1,m}$ and $N_{2,m}$ in Eqn. 3.12, whereas the second dimension runs along m, accounting for the $(l-1)$ genetic intervals. Once all the exchanges are processed and the marginal probabilities computed, the elements in the third dimension are divided by their corresponding marginal probabilities and then added up across the dimension. This gives us the two dimensional structure *postCount* containing values of $N_{0,m}$,$N_{1,m}$ and $N_{2,m}$ for all the genetic intervals. The new value of c_i for each genetic interval is then computed using Eqn. 3.12, and the process iterates until convergence is reached. Despite being a recursive algorithm (exchanges are generated recursively), it suffers from the computational bottleneck of having to process a large number of exchanges i.e., 25^{l-1} for l loci. This problem is overcome using the proposed recursive linking algorithm.

3.8 The Proposed Recursive Linking Algorithm

Let the entire order of genetic markers be decomposed into segments of equal width (h), such that all the intervals are covered. Thus, for l genetic markers the number of segments s is given by the following equation :

$$s = \frac{(l-1)}{(h-1)} \tag{3.14}$$

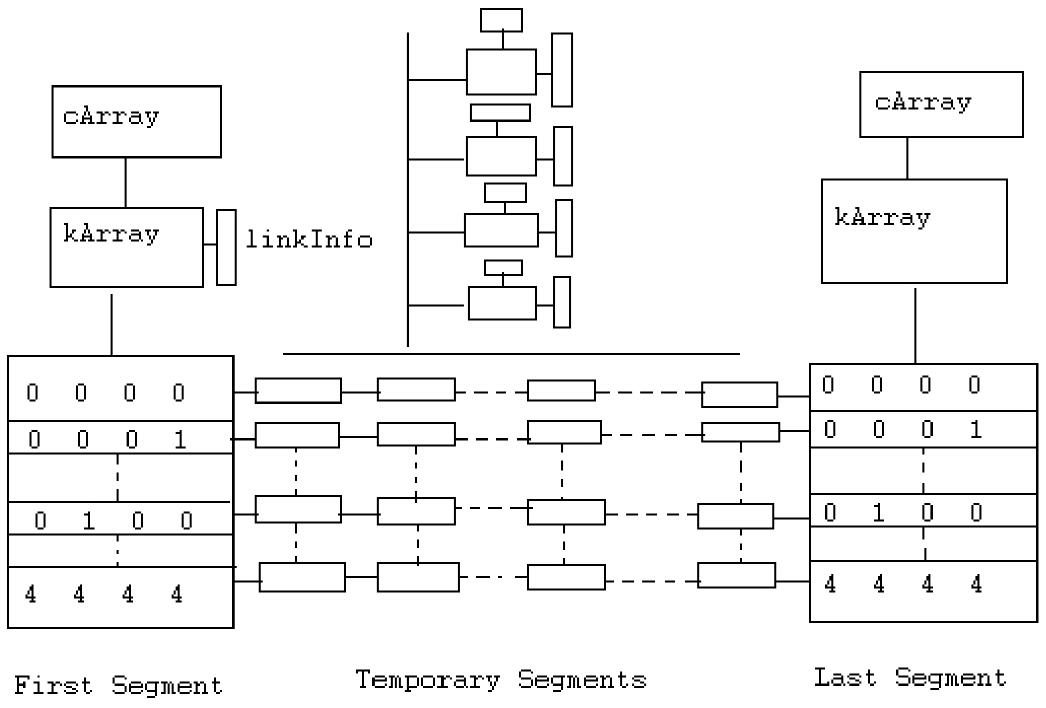


Figure 3.4: A visual depiction of the data structures

3.8.1 Overview of the Recursive Linking Algorithm

The first segment has an associated array called $kArrayFirst$ that contains all the exchanges for this segment. The array $linkInfoFirst$ stores the last row generated by the matrix R for each exchange of the segment. The array $cArrayFirst$ checks for each exchange and for each observed genotype of the first segment, which strands of the simulated tetrad (based on the model described in Eqn. 3.1, obtained via matrix R, match with the genotype. The matching status forms the last dimension of the array with length 4 and consists of symbols 1 and 0 indicating a match(1) or mismatch(0) respectively. For example, a matching status 1 0 0 1 for the first distinct genotype corresponding to the exchange pattern 0 0 2 3 4 in the first segment level indicates that among the 4 tetrads in meiosis generated by the exchange pattern 0 0 2 3 4 in the first segment, the observed genotype in question was found only on the first and the fourth tetrad. When we use this information over a combined segment formed using two segments, only a match at the same tetrad position will ensure a match for the combined segment.Note that R depends on exchange values on all intervals of the segment and its columns are sequentially dependent on each other with a lag of one. Similarly we have structures *kArrayLast* and *cArrayLast* for the last segment. Each exchange in the first segment branches out to $25^{(h-1)}$ exchanges in the following segment and creates $25^{2(h-1)}$ combined exchanges. This continues till the last segment is accounted for. In order to move along the segments following the model described in Eqn. 3.1, we need to know the last row (a tetrad pattern for the last locus of the segment) generated by the matrix R for the linked exchange of the previous segment corresponding to each combined exchange of the two segments. The following lemma states that only certain patterns are possible at the terminal locus of the adjoining segments. Hence we create arrays similar to *kArray*,$linkInfo$ and *cArray* for all the following segments between the first segment and the last segment and call them *kArrayTemp*[],$linkInfoTemp$[] and *cArrayTemp*[] respectively, where the array index denotes the segment numbers.

Lemma 3.1. *Under the model described by Eqn. 3.1 at any specified locus only one of the tetrad patterns 1 1 0 0, 0 1 1 0, 1 0 1 0, 1 0 0 1, 0 1 0 1 and 0 0 1 1 could occur.*

Proof: Recall that a tetrad pattern is an arrangement of the alleles of a particular gene in the simulated tetrad generated by a given exchange under the model described by Eqn. 3.1, where 1 and 0 indicate the parental alleles for the particular locus. Let P_1,P_2,$\cdots$,P_6 denote the tetrad patterns 1 1 0 0, 0 1 1 0, 1 0 1 0, 1 0 0 1, 0 1 0 1 and 0 0 1 1 respectively. The tetrad pattern at locus i for a exchange value s_i in the i^{th} genetic interval S_i is given by

$$T_{s_i} = f_k(f_j(T_{s_{i-1}}))$$

where ,$s_i = \{j,k\} \in S_i$ in Eqn. 3.2 and $f_k(.)$ and $f_j(.)$ are obtained from Eqn. 3.5. Note that $T_{s_0} = P_1$. In order to generate the patterns beginning with pattern T_{s_0} along with their sample points, (Table 3.2) we can see that all the sample points with the source pattern $T_{s_0} = P_1$ correspond to the patterns within $P_i(i = 1, \cdots, 6)$. Next we enumerate tetrad patterns beginning with source patterns $P_i(i = 1, 2, \cdots, 6)$. From Table 3.2 it is clearly seen that tetrad patterns cannot lie outside the set $P_i(i = 1, \cdots, 6)$. Hence the lemma is proved.

3.8.2 Counting Active Exchanges

Analogous to the function *kIsWorthy* in the straightforward algorithm we implement the concept of counting active exchanges for each segment. We call an exchange of a particular segment active if it has positive probability for at least one distinct genotype. Note that active exchanges will be different across segments as an active exchange depends on both, the observed genotypes (that vary across segments) and the tetrad pattern used in the generation of the matrix R. It is important to emphasize the substantial computational savings achieved by the elimination of the exchanges on a segment-wise basis in the proposed algorithm compared to the straightforward algorithm which eliminates exchanges one at a time. Elimination of a single exchange in the first segment has the effect of elimination of 25^{l-2} exchanges in the straightforward algorithm. In general, elimination of a single exchange in the i^{th} segment has the effect of eliminating 25^{l-i-1} exchanges in the straightforward algorithm. The following algorithm shows how active exchanges are computed.

Algorithm : Counting Active Exchanges

```
int startRowIndex,endRowIndex;
activeKFirst=0;
activeKLast = new int[6];
k=new int[height-1];
counter=new int[height-2];
startRowIndex=0;
endRowIndex=startRowIndex+height-1;
data1=GetData(startRowIndex,endRowIndex,0,(distinctGenotypes-1));
{Extracting genotype data for the first segment using data $trimdData2$}
startRowIndex=loci-height;
endRowIndex=loci-1;
data2=GetData(startRowIndex,endRowIndex,0,(distinctGenotypes-1));
{Extracting genotype data for the last segment using data $trimdData2$}
ActiveKLoopFirstAndLast(0,height);
```

Algorithm : ActiveKLoopFirstAndLast(0,height)

```
void ActiveKLoopFirstAndLast(int index,int height)
if index < height-2 then
 for counter[index]=0;counter[index]<24;counter[index]++ do
  ActiveKLoopLast(index+1,height);
 end for
end if
if index == height-2 then
 for int check1=0;check1<24;check1++ do
  for int check2=0;check2<height-2;check2++ do
   k[check2]=counter[check2];
  end for
  k[height-2]=check1;
  int[][]r;
  r=getRMatrix(k,firstCol);
  {$firstCol$ is 1 1 0 0}
  { FOR THE FIRST SEGMENT}
  if kIsWorthy(r,data1)==1 then
   activeKFirst++;
  end if
  { FOR THE LAST SEGMENT}
  for int i=0;i<6;i++ do
   r=getRMatrix(k,tempVec[i]);
   {$getRMatrix(.)$ computes Eqn. 3.6 for the $(i+1)^{th}$ tetrad pattern in Lemma 3.1}
   if kIsWorthy(r,data2)==1 then
    activeKLast[i]++;
   end if
  end for
 end for
```

end if

3.8.3 Computing *kArray*,*LinkInfo* and *cArray* for the First and the Last Segment

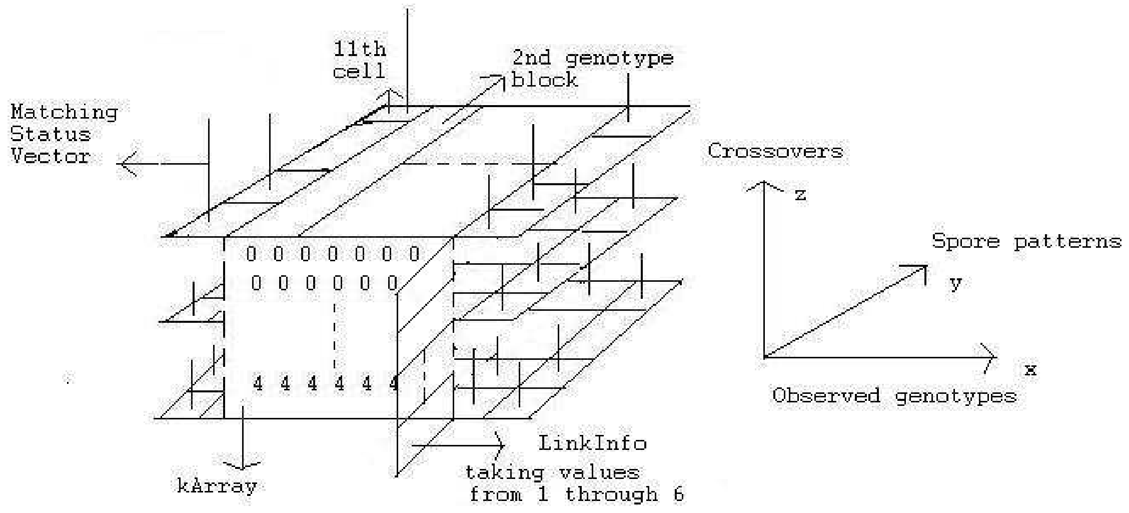


Figure 3.5: A visual depiction of the data structures *kArray*,*LinkInfo* and *cArray*

Figure 3.5 shows the data structures *kArray*,*linkInfo* and *cArray* as they are linked to each other on a particular segment. This structure is true for both the first and the last segments. As described in Section 3.8.1, the data structure *linkInfo* keeps track of the tetrad pattern which is one of the 6 possibilities described in Lemma 3.1. Structure *cArray* helps in computing the matching status over the combined segments, which in turn helps to compute the conditional probability of observed genotypes given the progressive consideration of exchanges as more and more segments are linked. This process is best understood by the algorithm shown below.

Algorithm : Computing *kArray*,*LinkInfo* **and** *cArray*

```
startRowIndex=0;
endRowIndex=height-1;
kCount1=0;
```

```
kCount = new int[6];
k=new int[height-1];
counter=new int[height-2];
kArrayFirst= new int[activeKFirst][height-1];
inkInfoFirst=new int[activeKFirst];
cArrayFirst=new int[activeKFirst][distinctGenotypes][4];
endRowIndex=height-1;
data1=GetData(startRowIndex,endRowIndex,0,(distinctGenotypes-1));
startRowIndex=loci-height;
endRowIndex=loci-1;
kArrayLast= new int[6][][];
cArrayLast= new int[6][][][];
for int i=0;i<6;i++ do
  kArrayLast[i]=new int[activeKLast[i]][height-1];
  cArrayLast[i]=new int[activeKLast[i]][distinctGenotypes][4];
end for
data2=GetData(startRowIndex,endRowIndex,0,(distinctGenotypes-1));
ComputeFirstAndLast(0,height);
```

Algorithm : ComputeFirstAndLast(0,height)

```
void LoopLast(int index,int height)
if index < height-2 then
  for counter[index]=0;counter[index]<24;counter[index]++ do
    LoopLast(index+1,height);
  end for
end if
if index == height-2 then
  for int check1=0;check1<24;check1++ do
    for int check2=0;check2<height-2;check2++ do
```

```
 k[check2]=counter[check2];
end for
k[height-2]=check1;
int[][]r;
{ THE FOLLOWING CODE COMPUTES OBJECTS FOR THE FIRST SEGMENT}
r=getRMatrix(k,firstCol);
if kIsWorthy(r,data1)==1 then
 kArrayFirst[kCount1]=setEqual(k);
 linkInfoFirst[kCount1]=Map6ToInt(r[r.length-1]);
 {Map6ToInt(.) converts the tetrad patterns in Lemma 3.1 to an integer between 1
 and 6}
 for int index1=0;index1<distinctGenotypes;index1++ do
  cArrayFirst[kCount1][index1]=MatchArray(r,getCol(data1,index1+1));
  {MatchArray(.) creates a matching status vector for the genotype and segment in
  question }
 end for
 kCount1++;
end if
{ THE FOLLOWING CODE COMPUTES OBJECTS FOR THE LAST SEGMENT}
for int i=0;i<6;i++ do
 r=getRMatrix(k,tempVec[i]); // r is segment-1 by 4
 if kIsWorthy(r,data2)==1 then
  kArrayLast[Map6ToInt(tempVec[i])][kCount[i]]=setEqual(k);
  for int index1=0;index1<distinctGenotypes;index1++ do
   cArrayLast[Map6ToInt(tempVec[i])][kCount[i]][index1]
   =MatchArray(r,getCol(data2,index1+1));
  end for
  kCount[i]++;
 end if
```

```
      end for
    end for
  end if
```

3.8.4 Computing *tempSumIndex*

Although not absolutely necessary in the implementation of the recursive linking algorithm, we compute a variable called *tempSumIndex* for the last segment which helps to speed up the EM iterations. Consider a combined exchange for all the segments. To compute Eqn. 3.7 i.e, to count the matches for the entire exchange we have to examine the matching status of all the segments and update them. To be considered a match for the entire segment at a particular position (out of 4 possible positions) one must have a match for all the segments at that position. Hence when the matching status values of two segments are updated, the resulting matching status is 1 if and only if both the segments have a value 1 at that position and 0 otherwise. This updated matching status is termed a *spore* in this paper. The distinction between a spore and matching status is that while a matching status is the original status of the segment, the spore is the matching status obtained after updating the matching status of all the previous segments. The following lemma restricts the number of possible spore patterns.

Lemma 3.2. *Under model described by Eqn. 3.1, for any exchange and any observed genotype the spore patterns 0 1 1 1, 1 0 1 1, 1 1 0 1, 1 1 1 0 and 1 1 1 1 are not possible.*

Proof: Recall that a spore pattern denotes the matching status of the tetrad of a diploid cell corresponding to an order of genetic markers. A 1 indicates a match, and 0 a mismatch of the observed genotype to the simulated tetrad for the exchange at hand, according to the model. Note that since each parent has a characteristic allele, it is not possible to have matches at 3 locations. Hence spore patterns with 3 or more matches are not possible. Hence the above lemma is proved.

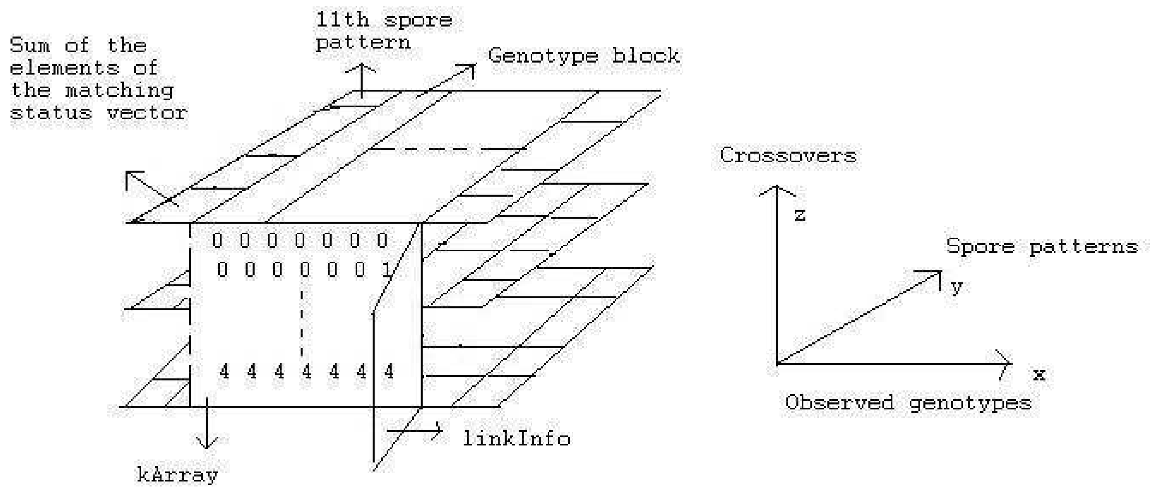


Figure 3.6: A visual depiction of structures *kArray*,*LinkInfo* and *cArray* of the last chromosome segment after computation of the sum of elements of the updated matching status vector

As a visual image of the interplay along different dimensions, we define a plane corresponding to each exchange. The blocks along the x dimension refer to distinct genotypes whereas its cells denote the sum of the updated matching status values for the 11 possible matching status values from the previous segment as specified by Lemma 3.2. The structure *tempSumIndex* essentially computes Eqn. 3.9 for the last segment except that the matching status value is computed for all possible tetrad patterns described in Lemma 3.1. Note that Lemma 3.1 and Lemma 3.2 restrict the array size and ensure efficient memory usage. The following algorithm illustrates the computation of *tempSumIndex*.

```
**Algorithm : Computing** *tempSumIndex*
int[][] spores=getSpores();
{getSpores(.) creates all the 11 possible spores (out of 16) using Lemma 3.2}
tempSumIndex = new int[6][][][];
for int i=0;i<6;i++ do
```

```
  tempSumIndex[i]=new int[distinctGenotypes][11][activeKLast[i]];
end for
for int index0=0;index0<6;index0++ do
  for int index1=0;index1<distinctGenotypes;index1++ do
    for int index2=0;index2<11;index2++ do
      for int index3=0;index3<activeKLast[index0];index3++ do
        tempSumIndex[index0][index1][index2][index3]
        =Sum(UpdateSpore(cArrayLast[index0][index3][index1],spores[index2]));
        {$UpdateSpore(.)$ updates the matching status as described earlier}
        {$Sum(.)$ adds up the elements of its argument}
      end for
    end for
  end for
end for
```

3.8.5 Computing the Recursive Structures *tempSum* and *tempPCount*

The structure *tempSumIndex* created for the last segment is used in conjunction with certain operations performed on each plane to compress the information for the last segment. Each exchange has a positive probability (since inactive exchanges have been omitted) defined in Eqn. 3.10 which is same for all the cells in the attached plane. Cells in a plane refer to the conditional probability up to a constant(divisor 4) for all possible situations. If we multiply cells by the exchange probability of the plane and sum across all the planes we essentially implement Eqn. 3.8 except that we do it for various matching status configurations in the previous segment. This allows us to implement Eqn. 3.8 for the combined segment as if the combined segment were never decomposed into segments. Note that all these operations are performed for a specific value of θ. We need to implement Eqn. 3.12 to update θ via the EM algorithm. This is done using the *pipes* shown in Figure 3.7. A cell on a plane has as many pipes as there are genetic intervals and each pipe has 3 associated categories

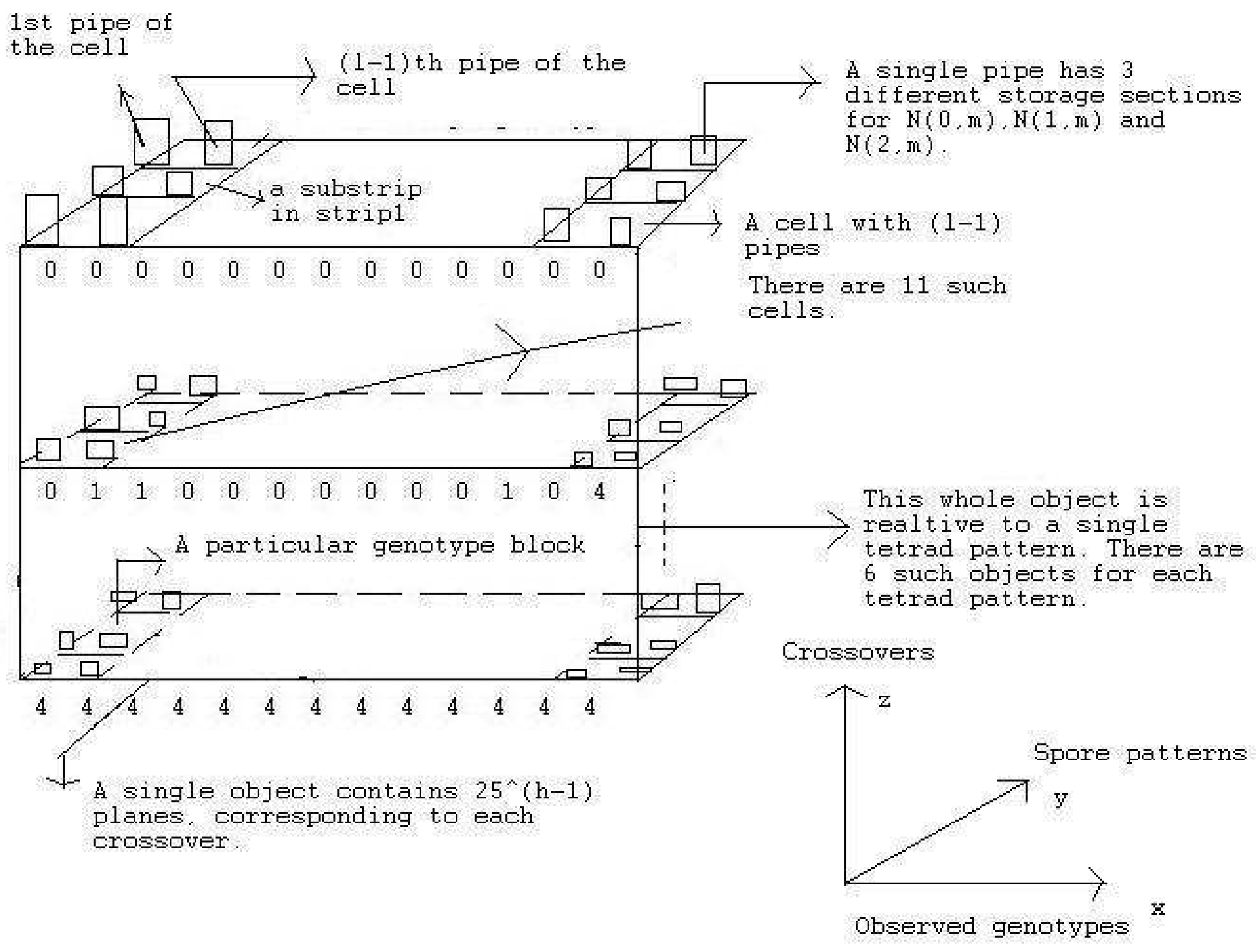

1st pipe of the cell
(l-1)th pipe of the cell
A single pipe has 3 different storage sections for N(0,m),N(1,m) and N(2,m).
a substrip in strip1
A cell with (l-1) pipes
There are 11 such cells.
0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 0 0 0 0 0 1 0 4
This whole object is realtive to a single tetrad pattern. There are 6 such objects for each tetrad pattern.
A particular genotype block
Crossovers
z
4 4 4 4 4 4 4 4 4 4 4 4 4
Spore patterns
y
A single object contains 25^(h-1) planes, corresponding to each crossover.
Observed genotypes
x

Figure 3.7: A visual depiction of structures *tempSum* and *tempPCount* of the last chromosome segment

for computing $N_{0,m}$,$N_{1,m}$ and $N_{2,m}$ in Eqn. 3.12. Note that as planes are added up (compressed) each exchange is categorized within its pipes according to its exchange values in the genetic intervals. This logic is similar to that underlying the implementation of *pCount* in the straightforward algorithm. The structures *tempSum* and *tempPCount* together constitute the recursive structure of the last segment. In the following pseudocode, structures *tempSum* and *tempPCount* implement Eqns. 3.8 and 3.12 respectively.

Algorithm : Computing the Recursive Objects *tempSum* **and** *tempPCount*

```
double[][][] FirstTempSum=new double[6][distinctGenotypes][11];
double[][][][][] FirstTempPCount=new double[6][distinctGenotypes][11][3][loci-1];
double[] lastCProb=getCProb(cProbOld,0,height-1)
{$getCProb$(.) extracts the current segment from the exchange probability $cProbOld$ }
for int index0=0;index0<6;index0++ do
 for int index1=0;index1<distinctGenotypes;index1++ do
  for int index3=0;index3<11;index3++ do
   for int index2=0;index2<activeKLast[index0];index2++ do
    double tempDouble=kProb(lastCProb,kArrayLast[index0][index2]);
    {$kProb$(.) computes the exchange probability for the
    exchange value in $kArrayLast[index0][index2]$}
    double condProb=tempSumIndex[index0][index1][index3][index2]/4.0;
    FirstTempSum[index0][index1][index3]+=condProb*tempDouble;
    for int index=loci-height;index<loci-1;index++ do
     FirstTempPCount[index0][index1][index3]
     [CellSpecial(kArrayLast[index0][index2]
     [index-loci+height])][index]+=condProb*tempDouble;
    end for
   end for
  end for
 end for
end for
```

3.8.6 Traversing the Segments by Updating the Recursive Structure

As we traverse the segments, for each segment we create the associated structures *kArray* (which is the same for all segments if equal width segments are assumed), *linkInfo* and *cArray*, by first computing the active exchanges and then computing the structures in a similar manner as described in Sections 3.8.2 and 3.8.3. Given a recursive structure for the last segment (on the right in Figure 3.8) the goal is to generate a recursive structure

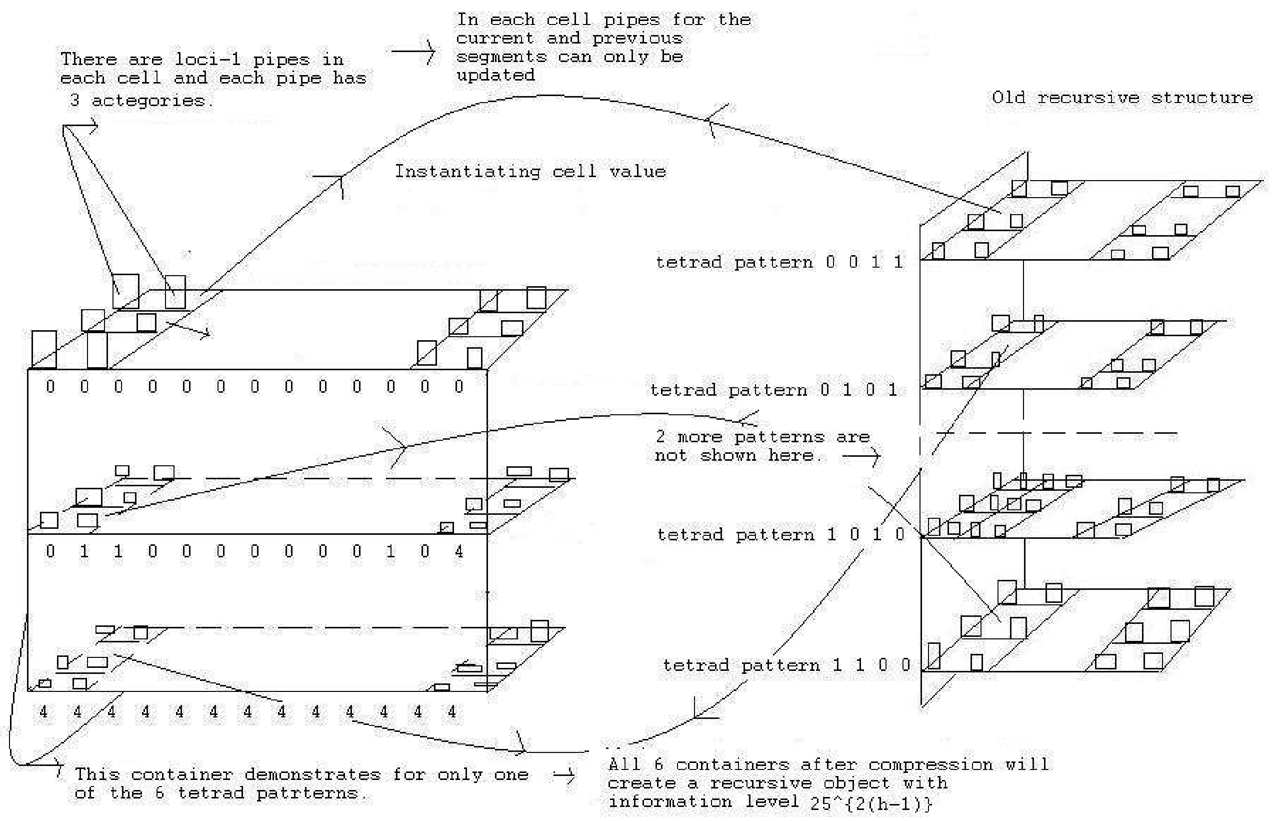

In each cell pipes for the current and previous segments can only be updated
There are loci-1 pipes in each cell and each pipe has 3 actegories.
Old recursive structure
Instantiating cell value
tetrad pattern 0 0 1 1
tetrad pattern 0 1 0 1
2 more patterns are not shown here.
tetrad pattern 1 0 1 0
tetrad pattern 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 0 0 0 0 0 1 0 4
4 4 4 4 4 4 4 4 4 4 4 4 4
This container demonstrates for only one of the 6 tetrad patrterns.
All 6 containers after compression will create a recursive object with information level 25^{2(h-1)}

Figure 3.8: A visual depiction of the updating of the recursive structure

for the second to last segment using the first one. This process is outlined in Figure 3.8. This structure has the property that $25^{(h-1)}$ exchanges have already been processed in a form such that Eqn. 3.12 can be implemented and any spore generated from the previous segment can be handled. Note that when an exchange from the previous segment(the second to last segment) is processed(allowing for all possible spore patterns of the previous segment), the corresponding spore is computed, its equivalent cell on the last segment is identified and $25^{(h-1)}$ combined exchanges of these two segments are processed. After all the exchanges from the previous segment are processed, the proposed algorithm generates a recursive structure that is exactly the same as that of the last segment without adding to the storage requirement. The recursive structure for the last but one segment now has $25^{2(h-1)}$ exchanges processed within it with provision for all possible spores from its preceding segment. This very feature shows how we can geometrically increase the information base (in terms of exchanges) of the "table look up " procedure and avoid traversing all the exchanges one at a time. One of the reasons this procedure works is because if we examine the combined exchanges of two segments, we see that exchange values for the left segment change only once for every $25^{(h-1)}$ combined exchanges. This allows us to delay the probability value updates when linking the segments. In the next few paragraphs we discuss in detail the updating procedure.

The structure on the left in Figure 3.8, called a *container*, is essentially a recursive structure except that has not been yet instantiated. We choose a cell on the plane corresponding to an exchange in the *container*. Note that the cell that is visited depends on the matching status of the *container*'s preceding segment and the *genotype* of the cell. The matching status of the cell in the *container* is updated with the matching status of the genotype corresponding to the segment of the *container* and a cell for the updated matching status is identified in the recursive structure (on the right in Figure 3.8), for that particular genotype.

The structure *tempSum* for the *container* is instantiated by multiplying the structure *tempSum* of the recursive structure with the exchange probability of the plane in *container*. Here, the term multiplication is used in a loose sense of multiplication. Note that a single multiplication operation processes 25^{h-1} exchanges at a time. We have $l-1$ pipes in each cell, and for the cell under consideration, the pipes are updated in the following manner. The pipes for the segments other than the segment of the *container* and following segments are left unfilled. The pipes for the *container*'s segment are filled for the categories (one of the 3) corresponding to the exchange values of each plane by multiplying the structure *tempSum* of the recursive structure with the plane probability (i.e., the exchange probability for the plane in question). For the pipes corresponding to the following segments (segments that have been processed already), all the pipes in the identified cell are copied and multiplied by the plane probability. This process is carried out for all the cells on all the planes of a *container*. The other 5 containers corresponding to the tetrad patterns in Lemma 3.1 are processed similarly. All of the planes for all the 6 containers are compressed as described earlier to generate a recursive structure for the segment corresponding to the *container*. Note that $\pi_{k|j}$ in Eqn. 3.12 is computed only up to a constant value. The denominator p_j

cannot be computed at the same time since that also involves traversing all exchange paths and is handled by the structure *tempSum* in the pseudocode. It is important to note that this creates no problem as at the end of the processing, the value of p_j is adjusted. The same issue arises in the implementation of the straightforward algorithm discussed earlier.The process is best understood by examining the pseudocode below and noticing how the arrays *tempSum* and *tempPCount* are updated in the recursive linking process.

```
**Algorithm : Updating the Recursive Structure**
double[][][] temp1TempSum = FirstTempSum;
double[][][][][] temp1TempPCount = FirstTempPCount;
double[][][] temp2TempSum=new double[6][distinctGenotypes][11];
double[][][][][] temp2TempPCount=new double[6][distinctGenotypes][11][3][loci-1];
**INTERMEDIATE STEPS**
int startRowIndex=0;
int endRowIndex=loci-height;//note that there is a common gene in each interval
int startPos =0;
int endPos=loci-height;
**for** int indexS=0;indexS<segments-2;indexS++ **do**
  startPos=endPos-height+2;
  double[] firstCProb=ChopAtoB(cProbOld,startPos,endPos);
  {$ChopAtoB(.)$ extracts elements in $cProbOld$ from $startPos$ to $endPos$}
  int[][] spores = getSpores();
  {Setting up the current temporary segment}
  kCount = new int[6];
  k=new int[height-1];
  counter=new int[height-2];
  activeKTemp = new int[6];
  **Get the relevant data**
  startRowIndex=endRowIndex-height+1; // endRowIndex is changed in the bottom.
  data=GetData(startRowIndex,endRowIndex,0,(distinctGenotypes-1));
```

```
Calculate active exchanges. Stored in activeKTemp.
ActiveKLoopTemp(0,height,indexS); // note this uses the object *data*.
{*ActiveKLoopTemp*(.) is similar in function to the previously
explained function *ActiveKLoopFirstAndLast*(.) except being for an intermediate segment}
**Allocate cells**
cArrayTemp = new int[6][][][];
linkInfoTemp = new int[6][];
kArrayTemp = new int[6][][];
**for** int i=0;i<6;i++ **do**
  cArrayTemp[i] = new int[activeKTemp[i]][distinctGenotypes][4];
  linkInfoTemp[i] = new int[activeKTemp[i]];
  kArrayTemp[i] = new int[activeKTemp[i]][height-1];
**end for**
{Run the loop to set *kArrayTemp*,*cArrayTemp* AND *linkInfoTemp*}
LoopTemp(0,height,indexS); // note this uses the object *data*
{*LoopTemp*(.) is similar in function to the function *ComputeFirstAndLast*(.) except
being for an intermediate segment}
{**Linking at work**}
**for** int index0=0;index0<6;index0++ **do**
  **for** int index1=0;index1<distinctGenotypes;index1++ **do**
    **for** int index2=0;index2<11;index2++ **do**
      **for** int indexK=0;indexK<activeKTemp[index0];indexK++ **do**
        int[] k=setEqual(kArrayTemp[index0][indexK]);
        double prob=kProb(firstCProb,k);
        int index01=linkInfoTemp[index0][indexK];
        int spike=getSporeMatch(UpdateSpore(cArrayTemp[index0][indexK][index1],
        spores[index2]));
```

```
          {$getSporeMatch(.)$ converts spores in Lemma 3.2 to one of the integers from 1 to
          11}
          temp2TempSum[index0][index1][index2]+=temp1TempSum[index01]
          [index1][spike]*prob;
          **for** int index=startPos-1;index<endPos;index++ **do**
            temp2TempPCount[index0][index1][index2]
            [CellSpecial(k[index-startPos+1])][index]+=
            temp1TempSum[index01][index1][spike]*prob;
          **end for**
          **for** int i=0;i<3;i++ **do**
            **for** int index=endPos;index<loci-1;index++ **do**
              temp2TempPCount[index0][index1][index2][i][index]+=
              temp1TempPCount[index01][index1][spike][i][index]*prob;
            **end for**
          **end for**
        **end for**
      **end for**
    **end for**
  **end for**
  **CHANGE TEMP1 TO TEMP2**
  temp1TempSum = setEqual(temp2TempSum);
  temp1TempPCount = setEqual(temp2TempPCount);
  **SET TEMP2 TO ZEROS FOR SECURITY**
  temp2TempSum=new double[6][distinctGenotypes][11];
  temp2TempPCount=new double[6][distinctGenotypes][11][3][loci-1];
  endPos=startPos-1;
  endRowIndex=startRowIndex;
**end for**
```

3.8.7 Linking with the First Segment

Linking with the first segment is the last step of the likelihood computation process. In this phase we do not have any previous matching status vectors and instead of *tempPCount* we have the structure *pCount* as in the straightforward algorithm. Note that the length of the first segment must be adjusted to account for both, even and odd number of genetic markers. That entails a trivial modification of the algorithm, and hence we do not mention the details. In the interest of clarity of description of the algorithm, we have assumed all the segments to be of equal length, and hence both an even and odd number of markers cannot be implemented without first changing the length of at least one segment; preferably that of the first one.

Algorithm : Linking with the First Segment

```
double[] firstCProb=getCProb(cProbOld,1,height-1);
for int index1=0;index1<activeKFirst;index1++ do
  int[] k=setEqual(kArrayFirst[index1]);
  double prob=kProb(firstCProb,k);
  int[] spike=new int[distinctGenotypes];
  {Spike is updated spore before the last segment}
  for int i=0;i<distinctGenotypes;i++ do
    spike[i]=getSporeMatch(cArrayFirst[index1][i]);
  end for
  {probFOld and pCount are global variables}
  int index0=linkInfoFirst[index1];
  for int index3=0;index3<distinctGenotypes;index3++ do
    probFOld[index3]+=temp1TempSum[index0][index3][spike[index3]]*prob;
    for int index=0;index<height-1;index++ do
      pCount[CellSpecial(k[index])][index][index3]+=
      temp1TempSum[index0][index3][spike[index3]]
      *prob*genotypeFreq[index3]/totalObs;
    end for
    for int i=0;i<3;i++ do
      for int index=height-1;index<loci-1;index++ do
        pCount[i][index][index3]+=
        temp1TempPCount[index0][index3][spike[index3]]
        [i][index]*prob*genotypeFreq[index3]/totalObs;
      end for
    end for
  end for
end for
```

3.8.8 Implementing the EM Algorithm by Recursive Linking

The following algorithm wraps around the entire EM algorithm as implemented in the proposed recursive linking algorithm. This is similar in many aspects to the straightforward algorithm except that the function *SpiralUp*() joins the broken segments and updates the exchange probability (c) estimate and treats the entire process as if it were never decomposed. The description of the algorithm is as follows:

Algorithm : Implementing the EM Algorithm by Recursive Linking

```
for int iteration=0;iteration<iterationLimit;iteration++ do
  double[] cProbNew=new double[order.length-1];
  probFOld=new double[distinctGenotypes];
  pCount=new double[3][order.length-1][distinctGenotypes];
  double[][] postCount=new double[3][order.length-1];
  RECURSIVELY JOINING THE BROKEN GENETIC SEGMENTS AND
  IMPLEMENTING THE EM ALGORITHM
  SpiralUp();
  { SpiralUp() Implements the following algorithms previously outlined in the order they
  appear:
  Algorithm1 : Computing the Recursive Objects tempSum and tempPCount
```

```
 Algorithm2 : Updating the Recursive Structure
 Algorithm3 : Joining with the First Segment }
 **Reporting the likelihood**
 logLikelihood=0.0;
 for int index=0;index<distinctGenotypes;index++ do
   logLikelihood+=genotypeFreq[index]*Math.log(probFOld[index]);
 end for
 System.out.println("The log-likelihood is "+logLikelihood+" at iteration "+iteration);
 GETTING THE POSTERIOR COUNTS :POSTCOUNT OF SIZE 3
 for int index1=0;index1<3;index1++ do
   for int index2=0;index2<order.length-1;index2++ do
     for int index3=0;index3<distinctGenotypes;index3++ do
       pCount[index1][index2][index3]=pCount[index1][index2][index3]/probFOld[index3];
     end for
     postCount[index1][index2]=Sum(pCount[index1][index2]);
   end for
 end for
 Getting the c probability estimates
 for int index1=0;index1<loci-1;index1++ do
   cProbNew[index1]=
   (postCount[1][index1]+2*postCount[2][index1])/(2.0*Sum(getCol(postCount,index1+1)));
 end for
 if Convergence(cProbOld,cProbNew,0.01) ==1 then
   System.out.println("Convergence Achieved !");
   break;
 end if
 cProbOld=setEqual(cProbNew);
end for
```

3.9 Time Complexity Comparison of the Algorithms

In the analysis of time complexity of the genetic map reconstruction algorithms [64], the running variable is l, the number of genetic markers. The core function of the algorithm is to process a large number of exchanges in real time. The algorithm would be required even if one wanted to compute just the likelihood for the initial exchange probabilities and not use the subsequent EM iterations. Hence in both the straightforward and the proposed algorithm we provide run time complexity analysis for the main computationally intensive phase, namely processing all the exchanges for a single iteration.

The loop in the straightforward algorithm runs for $25^{(l-1)}$ iterations. In each iteration the computation time associated with matrix R is $O(l)$, since matrix R has l columns and the computation time for each column is fixed. Since the observed genotypes are not known in advance we do a worst-case analysis for the computation of the vector *sum*. The worst case occurs when there is a match between an observed genotype and any of the 4 columns of the matrix R resulting in run time complexity of order $O(l)$. The vector *sum* has d_g elements and its size does not vary with l. Hence the total execution time for computing vector *sum* is $O(l)$. The vector addition involved in computing *pCount* entails the processing of d_g elements and thus accounts for run time complexity of $O(l)$. Hence the total run time complexity of the straightforward algorithm is given by

$$R_{st} = O\left(l25^{(l-1)}\right) \tag{3.15}$$

In the recursive linking algorithm the run time for each of the s segments is $O((h-1)25^{(h-1)}))$. So the total run time for the recursive algorithm is given by

$$R_{rl} = O\left(s(h-1)25^{(h-1)}\right) \tag{3.16}$$

which is minimized for $h = 3$ for any odd number of loci l. Hence for $h = 3$ the run time complexity for the recursive linking algorithm is of order $O(l)$ using Eqns. 3.14 and 3.16.

3.10 Run Time Results

We ran several jobs on a DELL PC (Model DM051 Pentinum(R) 4 CPU 3.40GHz and 4GB of RAM) for different number of genetic markers in their natural order of precedence on a data set from the linkage group-I of *Neurospora crassa* [46] for both algorithms. It was verified that both the algorithms provide the same likelihood for the same order of markers as they essentially solve the same problem but in different ways. The run time corresponds to the average time taken for a single EM iteration before convergence upon multiple starts. The resulting speedup is clearly evident from results in Table 3.3.

3.11 Additional Application of the Recursive Linking Algorithm

Besides computing the likelihood, the recursive linking algorithm can be used successfully to compute the standard error of c_i for each genetic interval S_i. Without this algorithm, computation of a standard error suffers from the same computational bottleneck as the computation of the likelihood. In the following sections we briefly describe the theory used to compute a standard error and show in detail how the recursive linking algorithm is used to implement the theory. The theory in Section 3.11.1 has appeared in a separate paper [61] but is mentioned here briefly in order to make this paper self-contained and the algorithm more readable.

3.11.1 Computing the Standard Error of MLE

We use the *SEM* algorithm [44] to compute the standard errors of $\Theta = (c_m, m = 1, \cdots, l-1)$. The large sample variance-covariance matrix is given by (Eqn. 2.3.5 in [44])

$$V = I_{oc}^{-1} + \Delta V \tag{3.17}$$

where,

$$\Delta V = I_{oc}^{-1} D (I - D)^{-1}$$

and D is the matrix corresponding to the rate of convergence of EM and I_{oc} is defined as below:

$$I_{oc} = E\left[I_o\left(\theta|Y\right)|Y_{obs},\theta\right]\Big|_{\theta=\theta^*} \tag{3.18}$$

where, $I_o(\theta|Y)$ is the complete-data observed information matrix.

The EM algorithm described in Section 3.1 implicitly defines a mapping $\theta \rightarrow M(\theta)$ via Eqn. 3.12 from the parameter space of θ,$(0,1]^{l-1}$, to itself such that

$$\theta^{(t+1)} = M(\theta^{(t)}), \qquad \text{for } t = 0, 1, \cdots$$

Since $M(\theta)$ is continuous and $\theta^{(t)}$ converges to the MLE θ^*(using EM algorithm), then θ^* must satisfy

$$\theta^* = M(\theta^*)$$

Therefore in the neighborhood of θ^*, using a Taylor series expansion, we get

$$\theta^{(t+1)} - \theta^* \approx (\theta^{(t)} - \theta^*)D$$

where,

$$D = \left(\frac{\partial M_j(\theta)}{\partial \theta_i}\right)\Big|_{\theta=\theta^*}$$

is the $(l-1)\times(l-1)$ Jacobian matrix for $M(\theta) = (M_1(\theta), \cdots, M_{l-1}(\theta))$ evaluated at $\theta = \theta^*$.

3.11.2 Computation of D

Let d_{ij} be the $(i,j)^{th}$ element of D and define $\theta^{(t)}(i)$ to be

$$\theta^{(t)}(i) = \left(\theta_1^*, \cdots, \theta_i^{(t)}, \theta_{i+1}^*, \cdots, \theta_{l-1}^*\right) \tag{3.19}$$

that is, only the i^{th} component in $\theta^{(t)}(i)$ is active in the sense that the other components are fixed at their MLE values. By the definition of d_{ij}, we have

$$\begin{aligned} d_{ij} &= \frac{\partial M_j(\theta^*)}{\partial \theta_i} \\ &= \lim_{\theta_i \to \theta_i^*} \frac{M_j\left(\theta_1^*, \cdots, \theta_{i-1}^*, \theta_i, \theta_{i+1}^*, \cdots, \theta_{l-1}^*\right) - M_j\left(\theta^*\right)}{\theta_i - \theta_i^*} \\ &= \lim_{\theta_i \to \theta_i^*} \frac{M_j\left(\theta^{(t)}(i)\right) - \theta_j^*}{\theta_i - \theta_i^*} \\ &= \lim_{t\to\infty} d_{ij}^{(t)} \end{aligned}$$

The following steps are performed to compute the d_{ij}'s.

INPUT: θ^* and $\theta^{(t)}$.

Step 1. Run the usual E and M steps to obtain $\theta^{(t+1)}(i)$.

Repeat steps 2-3 for $i = 1, \cdots, l-1$.

Step 2. Calculate $\theta^{(t)}(i)$ from Eqn. 3.19, and treating it as the current estimate of θ, run an additional iteration of EM to obtain $\theta\ ^{(t+1)}(i)$.

Step 3. Obtain the ratio

$$d_{ij} = \frac{\theta_j^{\ (t+1)}(i) - \theta_j^*}{\theta_i - \theta_i^*}, \qquad \text{for } j = 1, \cdots, l-1$$

OUTPUT: $\theta^{(t+1)}$ and $\{d_{ij}^{(t)}, i, j = 1, \cdots, l-1\}$.

We obtain $d_{ij}$when the sequence $d_{ij}^{(t^*)}, d_{ij}^{(t^*+1)}, \cdots$ is stable for some t^*. This process may result in using different values of t^* for different d_{ij} elements.

3.11.3 Evaluation of I_{oc}^{-1}.

The complete-data information for the $(i,j)^t h$ element ($i = 1, \cdots, l-1$ and $j = 1, \cdots, l-1$) is given by:

$$\begin{aligned} I^o(i,j) &= -\frac{\partial^2 f^c(x,\theta)}{\partial\theta_i \partial\theta_j} \\ &= -\sum_k x_k \frac{\pi_k \frac{\partial^2}{\partial\theta_i\partial\theta_j}(\pi_k) - \frac{\partial}{\partial\theta_i}(\pi_k)\frac{\partial}{\partial\theta_j}(\pi_k)}{\pi_k^2} \end{aligned}$$

The complete-data information for the $(i,j)^{th}$ element ($i = 1, \cdots, l-1$ and $j = 1, \cdots, l-1$) is given by:

$$\begin{aligned} I^o(i,j) &= -\frac{\partial^2 f^c(x,\theta)}{\partial\theta_i \partial\theta_j} \\ &= -\sum_k x_k \frac{\pi_k \frac{\partial^2}{\partial\theta_i\partial\theta_j}(\pi_k) - \frac{\partial}{\partial\theta_i}(\pi_k)\frac{\partial}{\partial\theta_j}(\pi_k)}{\pi_k^2} \end{aligned}$$

Using Eqn. 3.18,

$$
\begin{aligned}
I_{oc} &= E\left[I^o(\theta|Y)\,|Y_{obs},\theta\right]\Big|_{\theta=\theta^*} \\
&= \sum_k \left[-x_k \frac{\pi_k \frac{\partial^2}{\partial\theta_i\partial\theta_j}(\pi_k) - \frac{\partial}{\partial\theta_i}(\pi_k)\frac{\partial}{\partial\theta_j}(\pi_k)}{\pi_k^2}\left[E\left(x_k\Big|n,\theta^*\right)\right]\right] \\
&= \sum_k \left[-x_k \frac{\pi_k \frac{\partial^2}{\partial\theta_i\partial\theta_j}(\pi_k) - \frac{\partial}{\partial\theta_i}(\pi_k)\frac{\partial}{\partial\theta_j}(\pi_k)}{\pi_k^2}\left[\sum_{j'=1}^{N} n'_j \pi_{k|j'}(\theta^*)\right]\right] \\
&= \sum_{j'=1}^{N} \frac{n_{j'}}{p_{j'}}\left[\sum_k \left[\frac{\pi_{j'|k}}{\pi_k}\frac{\partial}{\partial\theta_i}(\pi_k)\frac{\partial}{\partial\theta_j}(\pi_k) - \pi_{j'|k}\frac{\partial^2}{\partial\theta_i\partial\theta_j}(\pi_k)\right]\right] \text{[using Eqn. 3.12]} \qquad (3.20)
\end{aligned}
$$

Note that this is the only part in the computation of standard error that requires processing an exponential number of exchanges and the proposed recursive linking algorithm can be successfully used to address the computational bottleneck.

3.11.4 Algorithm to Compute the Standard Error of MLE

In the following algorithm active exchanges are computed and the cells allocated in a similar manner as in the likelihood computation. Here also we start with the last segment with all suitable indices as before and attempt to compute Eqn. 3.20. The structure *probDeri* computes the double partial derivatives while *probSingle* and *probSingle2* compute the univariate derivatives in Eqn. 3.20. The structure *FirstTempInfo* accounts for the conditional probability term along with the double derivative whereas the structure *FirstTempInfo2* accounts for the conditional probability term along with the product of the univariate derivatives. Notice that the structures *FirstTempInfo* and *FirstTempInfo2* will be recursively created for all the segments as they progressively help to move across all the exchanges as in the likelihood computation process.

Compute Active Exchanges and do Cell Allocations for all the Segments

{ For each segment the structures $kArray$,$linkInfo$ and $cArray$ are realized using similar algorithms as in Section 3.8.2 and Section 3.8.3. The structures $kArray$,$linkInfo$ and $cArray$ for the intermediate segments are arrays $kArrayTemp[]$,$linkInfoTemp[]$ and $cArrayTemp[]$, where the array index indicates a segment as earlier mentioned in the algorithm in Section 3.8.6.

```
}
Last Part First
double[] lastCProb=ChopAtoB(cProbFinal,loci-height+1,loci-1);
double probDeri=0.0,probSingle=0.0,probSingle2=0.0;
double[][][][][] FirstTempInfo=new double[6][11][height-1][height-1][distinctGenotypes];
double[][][][][] FirstTempInfo2=new double[6][11][height-1][height-1][distinctGenotypes];
double[][][][] FirstTempSingleD=new double[6][11][height-1][distinctGenotypes];
double[][][] FirstTempSum=new double[6][distinctGenotypes][11];
for int index1=0;index1<6;index1++ do
 for int index6=0;index6<activeKLast[index1];index6++ do
  double prob=kProb(lastCProb,kArrayLast[index1][index6]);
  for int index2=0;index2<11;index2++ do
   for int index5=0;index5<distinctGenotypes;index5++ do
    condProb=Sum(UpdateSpore(cArrayLast[index1][index6][index5],spores[index2]))/4.0;
    for int index3=loci-height;index3<loci-1;index3++ do
     probSingle=kProbSingle(lastCProb,kArrayLast[index1][index6],index3-loci+height+1);
     for int index4=loci-height;index4<loci-1;index4++ do
      if  index3! = index4 then
       probDeri=kProbDeriOff(lastCProb,kArrayLast[index1][index6],
           index3-loci+height+1,index4-loci+height+1);
      else
       probDeri=kProbDeriDiag(lastCProb,kArrayLast[index1][index6],
           index3-loci+height+1);
      end if
      FirstTempInfo[index1][index2][index3-loci+height]
          [index4-loci+height][index5]+=condProb*probDeri;
      probSingle2=kProbSingle(lastCProb,kArrayLast[index1]
          [index6],index4-loci+height+1);
      FirstTempInfo2[index1][index2][index3-loci+height]
```

```
                [index4-loci+height][index5]+=condProb/prob*probSingle*probSingle2;
           end for
           FirstTempSingleD[index1][index2][index3-loci+height]
                 [index5]+=condProb*probSingle;
         end for
         FirstTempSum[index1][index5][index2]+=condProb*prob;
       end for
     end for
   end for
 end for
```

Let us consider how the information matrix is configured when the second to last segment is processed. Note that the dimension of the information matrix (which is inherently a square matrix) has now increased from $height - 1$ to $2(height - 1)$ to account for intervals of 2 segments. We have 4 different cases for rows and columns arising from these 2 segments. For each of these cases, the double derivatives and single derivatives are processed individually. For the first case, the rows and columns are both obtained from the second to last segment; for the second case, the rows are obtained from the second to last segment and the columns from the last segment and so on, until for the fourth case, both the rows and columns are obtained from the last segment. Notice in the pseudocode below how double derivative computation is necessary only for the first case and for the remaining cases everything can be computed from the previous segment except for computation of the single derivatives. The probability adjustments are made in a similar manner as during the likelihood computation process.

```
Recursive Loop across all the Segments
startRowIndex=0;
endRowIndex=loci-height;//note that there is a common gene in each interval
int startPos =0;
int endPos=loci-height;
```

```
double[][][][][]temp1Info=FirstTempInfo;
double[][][][][]temp1Info2=FirstTempInfo2;
double[][][][]temp1SingleD=FirstTempSingleD;
double[][][]temp1Sum=FirstTempSum;
for int indexS=0;indexS<segments-2;indexS++ do
 for int index1=0;index1<6;index1++ do
  for int indexK=0;indexK<activeKTemp[index1];indexK++ do
   double prob=kProb(tempCProb,kArrayTemp[index1][indexK]);
   for int index2=0;index2<11;index2++ do
    for int index5=0;index5<distinctGenotypes;index5++ do
     int spike=getSporeMatch(UpdateSpore(cArrayTemp[index1][indexK]
         [index5],spores[index2]));
     int index01=linkInfoTemp[index1][indexK];
     temp2Sum[index1][index5][index2]+=temp1Sum[index01][index5][spike]*prob;
     for int index3=startPos-1;index3<loci-1;index3++ do
      for int index4=startPos-1;index4<loci-1;index4++ do
       if  (index3>=startPos-1 AND index3<=endPos-1)
           AND (index4>=startPos-1 AND index4<=endPos-1) then
        if index3 != index4 then
         probDeri=kProbDeriOff(tempCProb,kArrayTemp[index1]
             [indexK],index3-startPos+1+1,index4-startPos+1+1);
        else
         probDeri=kProbDeriDiag(tempCProb,kArrayTemp[index1]
             [indexK],index3-startPos+1+1);
        end if
        temp2Info[index1][index2][index3-startPos+1]
            [index4-startPos+1][index5]+=temp1Sum[index01]
               [index5][spike]*probDeri;
        probSingle=kProbSingle(tempCProb,kArrayTemp[index1]
      [indexK],index3-startPos+1+1);
  probSingle2=kProbSingle(tempCProb,kArrayTemp[index1]
      [indexK],index4-startPos+1+1);
  temp2Info2[index1][index2][index3-startPos+1]
      [index4-startPos+1][index5]+=temp1Sum[index01]
          [index5][spike]*probSingle*probSingle2/prob;
 **else**
  **if** (index3>=startPos-1 AND index3<=endPos-1)
      AND (index4>=endPos AND index4<=loci-2) **then**
   probSingle=kProbSingle(tempCProb,kArrayTemp[index1]
       [indexK],index3-startPos+1+1);
   temp2Info[index1][index2][index3-startPos+1]
       [index4-startPos+1][index5]+=temp1SingleD[index01]
         [spike][index4-endPos][index5]*probSingle;
   temp2Info2[index1][index2][index3-startPos+1]
       [index4-startPos+1][index5]+=temp1SingleD[index01]
         [spike][index4-endPos][index5]*probSingle;
  **else**
   **if** (index3>=endPos AND index3<=loci-2)
   AND (index4>=startPos-1 AND index3<=endPos-1) **then**
    probSingle=kProbSingle(tempCProb,kArrayTemp
      [index1][indexK],index4-startPos+1+1);
    temp2Info[index1][index2][index3-startPos+1]
      [index4-startPos+1][index5]+=temp1SingleD
      [index01][spike][index3-endPos][index5]*probSingle;
    temp2Info2[index1][index2][index3-startPos+1]
      [index4-startPos+1][index5]+=temp1SingleD
      [index01][spike][index3-endPos][index5]*probSingle;
   **else**
```

```
            if (index3>=endPos AND index3<=loci-2)
             AND (index4>=endPos AND index4<=loci-2) then
             temp2Info[index1][index2][index3-startPos+1]
             [index4-startPos+1][index5]+=temp1Info[index01]
                [spike][index3-endPos][index4-endPos]
                 [index5]*prob;
             temp2Info2[index1][index2][index3-startPos+1]
             [index4-startPos+1][index5]+=temp1Info2
             [index01][spike][index3-endPos][index4-endPos]
                [index5]*prob;
            end if
           end if
          end if
         end if
         if (index3>=startPos-1 AND index3<=endPos-1) then
          probSingle=kProbSingle(tempCProb,kArrayTemp[index1]
            [indexK],index3-startPos+1+1);
          temp2SingleD[index1][index2][index3-startPos+1][index5]+=
            temp1Sum[index01][index5][spike]*probSingle;
         else
          temp2SingleD[index1][index2][index3-startPos+1][index5]+=
            temp1SingleD[index01][spike][index3-endPos][index5]*prob;
         end if
        end for
       end for
      end for
     end for
    end for
   end for
```

```
  Change temp1 to temp2
  temp1Info = setEqual(temp2Info);
  temp1Info2 = setEqual(temp2Info2);
  temp1SingleD = setEqual(temp2SingleD);
  temp1Sum=setEqual(temp2Sum);
  endPos=startPos-1;
  endRowIndex=startRowIndex;
end for
```

Computation of double and single derivatives for all the segments completes with joining with the first segment as earlier.

```
Linking with the First Segment
double[] firstCProb=ChopAtoB(cProbFinal,1,height-1);
double[][][] tempInfo=new double[loci-1][loci-1][distinctGenotypes];
double[][][] tempInfo2=new double[loci-1][loci-1][distinctGenotypes];
double[][] tempSingleD=new double[loci-1][distinctGenotypes];
for int index1=0;index1<activeKFirst;index1++ do
  int[] spike=new int[distinctGenotypes];
  for int i=0;i<distinctGenotypes;i++ do
    spike[i]=getSporeMatch(cArrayFirst[index1][i]);
  end for
  prob=kProb(firstCProb,kArrayFirst[index1]);
  index01=linkInfoFirst[index1];
  for int index2=0;index2<distinctGenotypes;index2++ do
    probFOld1[index2]+=temp1Sum[index01][index2][spike[index2]]*prob;
    for int index3=0;index3<loci-1;index3++ do
      for int index4=0;index4<loci-1;index4++ do
        if (index3>=0 AND index3<=height-2)
            AND (index4>=0 AND index4<=height-2) then
```

```
 if index3 != index4 then
  probDeri=kProbDeriOff(firstCProb,kArrayFirst[index1],index3+1,index4+1);
 else
  probDeri=kProbDeriDiag(firstCProb,kArrayFirst[index1],index3+1);
 end if
 tempInfo[index3][index4][index2]+=temp1Sum[index01][index2]
   [spike[index2]]*probDeri;
 probSingle=kProbSingle(firstCProb,kArrayFirst[index1],index3+1);
 probSingle2=kProbSingle(firstCProb,kArrayFirst[index1],index4+1);
 tempInfo2[index3][index4][index2]+=temp1Sum[index01][index2]
   [spike[index2]]*probSingle*probSingle2/prob;
else
 if (index3>=0 AND index3<=height-2)
   AND (index4>=height-1 AND index4<=loci-2) then
  probSingle=kProbSingle(firstCProb,kArrayFirst[index1],index3+1);
  tempInfo[index3][index4][index2]+=temp1SingleD[index01][spike[index2]]
    [index4-endPos][index2]*probSingle;
  tempInfo2[index3][index4][index2]+=temp1SingleD[index01]
    [spike[index2]][index4-endPos][index2]*probSingle;
 else
  if (index3>=height-1 AND index3<=loci-2)
    AND(index4>=0 AND index4<=height-2) then
   probSingle=kProbSingle(firstCProb,kArrayFirst[index1],index4+1);
   tempInfo[index3][index4][index2]+=temp1SingleD[index01]
     [spike[index2]][index3-endPos][index2]*probSingle;
   tempInfo2[index3][index4][index2]+=temp1SingleD[index01]
     [spike[index2]][index3-endPos][index2]*probSingle;
  else
   if (index3>=height-1 AND index3<=loci-2)
```

```
          AND (index4>=height-1 AND index4<=loci-2) **then**
         tempInfo[index3][index4][index2]+=temp1Info[index01]
            [spike[index2]][index3-endPos][index4-endPos][index2]*prob;
         tempInfo2[index3][index4][index2]+=temp1Info2[index01]
            [spike[index2]][index3-endPos][index4-endPos][index2]*prob;
        end if
       end if
      end if
     end if
    end for
    if index3>=0 AND index3<=height-2 then
     probSingle=kProbSingle(firstCProb,kArrayFirst[index1],index3+1);
     tempSingleD[index3][index2]+=temp1Sum[index01][index2]
        [spike[index2]]*probSingle;
    else
     tempSingleD[index3][index2]+=temp1SingleD[index01]
        [spike[index2]][index3-endPos][index2]*prob;
    end if
   end for
  end for
end for
```

We compute the information matrix using Eqn. 3.20. Computing the variance-covariance matrix using Eqn. 3.17is straightforward. We do not show the computation of matrix D in Eqn. 3.17 represented by *dm2mat*in the pseudocode since the implementation is straightforward.

Getting the Expected Information Matrix

```
double[][] info=new double[loci-1][loci-1];
double[][] info1=new double[loci-1][loci-1];
double sumCov=0.0;
```

```
for int index1=0;index1<loci-1;index1++ do
 for int index2=0;index2<loci-1;index2++ do
  sumCov=0.0;
  for int index3=0;index3<distinctGenotypes;index3++ do
   sumCov+=(tempInfo2[index1][index2][index3]-tempInfo[index1][index2][index3])
        *genotypeFreq[index3]/probFOld1[index3];
  end for
  info[index1][index2]=sumCov;
 end for
end for
Matrix inverse=new Matrix(info).inverse();
Matrix dm2Mat =new Matrix(dm2);
Matrix delV=inverse.times(dm2Mat).times((Matrix.identity(loci-1,loci-1)
.minus(dm2Mat)).inverse());
Matrix varCov=inverse.plus(delV);
for int index=0;index<varCov.getRowDimension();index++ do
 SE[index]=Math.sqrt(varCov.get(index,index));
end for
```

3.12 Genetic Mapping from the RFLP data of *NEUROSPORA CRASSA*

We use our algorithm to create a genetic map for the entire genome of *Neurospora crassa.* The chromosomes involve large number of markers (around 60). The computation of the likelihood along with the estimates of c_i and their standard errors would have been impractical with any straightforward algorithm, however it was made possible with the proposed algorithm. We used the widely known stochastic optimization technique *simulated annealing* to determine the best genetic map by performing combinatorial optimization over the space of possible marker orders.

3.12.1 The Use of Simulated Annealing for Searching the Best Order.

Several combinatorial optimization problems can be tackled using simulated annealing (SA) as a stochastic optimization technique [35]. We use SA to determine the best order of genetic markers by sampling stochastically from the space of all possible gene orders. Simulated annealing has been used previously [17], to reconstruct chromosomes based on binary scoring of DNA fragments and a Hamming distance-based objective function. In our case, the objective function is the likelihood of a particular order of genes on a genetic map obtained upon convergence of the EM algorithm ([45], [67]). Thus, in our case a single computation of the objective function for a single order of markers is quite expensive. The SA provides a stochastic sampling technique to search for a good order as follows:

1. Choose a random order of probes, Π, and calculate $f(\Pi)$.

2. Choose a random segment within the ordering Π.

3. Perform a segment reversal and call the new ordering Π'.

4. Compute $f(\Pi')$.

5. If $f(\Pi')$ is less than $f(\Pi)$, then retain the new order. However, if $f(\Pi')$ is larger than $f(\Pi)$, then generate a random number between 0 and 1. If this random number is less than $E(-(f(\Pi') - f(\Pi))/T$, then retain the new order. Here T is the "temperature" of the annealing schedule.

6. Proceed to anneal the temperature in multiplicative steps, i.e., decrease T by a factor F at each SA iteration. Hold each value of T constant until D re-orderings have been attempted or S successful re-orderings have resulted, whichever comes first. If the number of successes equals zero for a given step, the process is complete; otherwise go to step 2.

Simulated annealing (SA) has been used to create a genetic map of chromosome-II in *Neurospora crassa*([34], [46]). For each order in SA, the likelihood is computed, and the order with the maximum value of the likelihood is selected as the desired map. The values of the parameters used in the search of the best order are $T = 10.0$,$F = 0.25$,$D = 500$ and $S = 50$. For the best order, c_is, their standard errors and their corresponding recombination fractions (see Section 3.6.1) for each genetic interval are computed. These statistics are reported for the best order in Table 3.4 (displayed at the end of the paper). The best order for linkage group-II turned out to be different from the published order [46] and has log-likelihood value of -63.9341. There are at least two reasons for the discrepancy. The simplest is that the maps were constructed by hand, so there is only a limited number of possible solutions considered. Metzenberg(personal communication) has also indicated that he held additional data that helped him decide on the map, data which the current computation did not have access to. The maps and their statistics for the remaining six chromosomes of *N. crassa* can be found at *http://gene.genetics.uga.edu*.

3.13 Conclusions

A multi-locus genetic likelihood was computed based on a mathematical model of the chromatid exchange in meiosis that accounts for any type of bivalent configuration in a genetic interval in any specified order of genetic markers. We proposed an algorithm that can reduce the time complexity of computations, for example, computing the likelihood, finding estimates of parameters (c_i) and their standard errors from exponential to linear time in the number of genetic markers. To illustrate the advantages of the proposed algorithm, we compared it alongside a straightforward algorithm presenting both, theoretical worst-case analysis and runtime results. Finally as an application, we reconstructed a genetic map of the entire genome of the model fungal system *Neurospora crassa* which otherwise would have been impractical to achieve, considering the number of markers involved.

ACKNOWLEDGEMENTS

The research was supported in part by a grant from the U.S. Dept. of Agriculture under the NRI Competitive Grants Program (Award No: GEO-2002-03590) to Drs. Bhandarkar and Arnold. We thank the College of Agricultural and Environmental Sciences, University of Georgia for their support.

Table 3.2: Possible Tetrad Patterns

Source Pattern	**Generated Patterns**	**Sample Points**
P_1	P_1	(0,0),(1,1),(2,2),(3,3),(4,4)
	P_2	(0,1),(1,0),(1,2),(1,4)
	P_3	(0,2),(2,0),(2,1),(2,3)
	P_4	(0,3),(3,0),(3,2),(3,4)
	P_5	(0,4),(4,0),(4,1),(4,3)
	P_6	(1,3),(3,1),(2,4),(4,2)
P_2	P_1	(0,1),(1,0),(2,1),(4,1)
	P_2	(0,0),(0,2),(2,0),(0,4),(4,0),(1,1),(2,2),(3,3),(4,4),(4,2),(2,4)
	P_3	(1,2),(3,4)
	P_4	(1,3),(3,1)
	P_5	(1,4),(3,2)
	P_6	(0,3),(3,0),(2,3),(4,3)
P_3	P_1	(0,2),(2,0),(1,2),(3,2)
	P_2	(2,1),(2,3),(4,3)
	P_3	(0,0),(0,1),(1,0),(0,3),(3,0),(1,1),(2,2),(3,3),(4,4),(1,3),(3,1)
	P_4	(4,1)
	P_5	(4,2),(2,4)
	P_6	(0,4),(4,0),(1,4),(3,4)
P_4	P_1	(0,3),(3,0),(2,3),(4,3)
	P_2	(1,3),(3,1)
	P_3	(1,4),(3,2)
	P_4	(0,0),(0,2),(2,0),(0,4),(4,0),(1,1),(2,2),(3,3),(4,4),(4,2),(2,4)
	P_5	(1,2),(3,4)
	P_6	(0,1),(1,0),(2,1),(4,1)
P_5	P_1	(0,4),(4,0),(1,4),(3,4)
	P_2	(4,1)
	P_3	(4,2),(2,4)
	P_4	(2,1),(2,3),(4,3)
	P_5	(0,0),(0,1),(1,0),(0,3),(3,0),(1,1),(2,2),(3,3),(4,4),(1,3),(3,1)
	P_6	(0,2),(2,0),(1,2),(3,2)
P_6	P_1	(1,3),(3,1),(4,2),(2,4)
	P_2	(0,3),(3,0),(3,2),(3,4)
	P_3	(0,4),(4,0),(4,1),(4,3)
	P_4	(0,1),(1,0),(1,2),(1,4)
	P_5	(0,2),(2,0),(2,3),(2,1)
	P_6	(0,0),(1,1),(2,2),(3,3),(4,4)

Table 3.3: Run time Comparison of Straight forward and Recursive Algorithm

Loci(l)	**Runtime(secs)**	
	Straight forward algorithm	**Recursive Linking Algorithm**
5	2.49	0.073
7	1336.45	0.298
9	> 43200.0	0.66
21	∞	5.61
41	∞	8.93
61	∞	13.03

Table 3.4: Best Genetic Map with Estimates of Parameters (c), its Standard Error(σ_c) and Recombination Fraction (r) on Linkage Group-II of *N.crassa*

Genes	c[a]	σ_c	r	**Genes**[b]	c	σ_c	r
Tel IIL	0	0.0022	0	Ncr-5	0.1588	0.2164	0.1462
hsps-1	0.2127	0.2019	0.1901	preg, cit-1	0.1524	0.2056	0.1408
00008	0.0002	0.0078	0.0002	12:11C, pma-1	0.3229	0.308	0.2708
pSK2-1A	0.1263	0.1709	0.1183	nuo78	0.0012	0.0248	0.0012
3:9A	0.1409	0.1894	0.131	21:D3	0.0012	0.0248	0.0012
Fsr-52	0.0007	0.0182	0.0007	X18:9A	0.0012	0.0248	0.0012
AP32b.1,R31.1, vph-1	0.1555	0.2268	0.1434	11:F3	0.1614	0.2196	0.1484
Fsr-32	0.0011	0.0238	0.0011	ccg-7	0.0012	0.0241	0.0012
Cen II, arg-5	0.0011	0.0238	0.0011	vma-2	0.0012	0.0241	0.0012
25:1D	0.0011	0.0238	0.0011	DB0001	0.0012	0.0241	0.0012
G8:11H	0.0011	0.0238	0.0011	Fsr-3	0.1588	0.2164	0.1462
atp-2	0.0011	0.0238	0.0011	X24:A11	0.1588	0.2164	0.1462
cya-4, Fsr-55	0.0011	0.0238	0.0011	eas=ccg-2=bli-7	0.1588	0.2164	0.1462
AP5i.1	0.0011	0.0238	0.0011	bli-4	0.1555	0.2119	0.1434
AP5.1	0.0011	0.0238	0.0011	Fsr-34	0.3153	0.3057	0.2656
AP3.2	0.154	0.196	0.1421	Fsr-17	0.001	0.0223	0.001
AP8u.4	0.154	0.196	0.1421	AP13.4 , 8:4GL	0.001	0.0223	0.001
H3H4	0.0011	0.0237	0.0011	AP32c.2, R32.2	0.001	0.0223	0.001
Ncr-2	0.0011	0.0237	0.0011	leu-6	0.1557	0.2129	0.1436
				Tel IIR			

[a] c, σ_c and r correspond to the genetic interval formed by the genes at the current row and the following row

[b] continuation of the first 4 columns of Table 3.4

Chapter 4

LIKELIHOOD-BASED INTEGRATED GENETIC AND PHYSICAL MAP OF *NEUROSPORA CRASSA* [1]

[1]Tewari,S., Bhandarkar, James Griffith, S.M. and Arnold, J. 2008
Journal of The American Statistical Association.
To be submitted.

4.1 Abstract

We model the two processes of clone/probe hybridizations to create a physical map and the chromatid exchange process to create a genetic map from a single cross. From this modeling framework a joint likelihood function is developed in which distances on each map and the order of markers on each map appear as parameters. The method of maximum likelihood is implemented to generate an integrated physical and genetic map at 23 kilobase pairs resolution for the model fungal system, *Neurospora crassa.* The integrated map permits the determination empirically of the mapping function relating recombination distance on the genetic map to physical distances in bp on the physical map. Some limitations of this mapping function are discussed in the light of the "centromere effect" on the physical size of a map unit on the genetic map and local variation in the physical size of a map unit over an order of magntude in *N. crassa.* In addition, the integrated maps permitted us to examine the non random distribution of repeated DNA sequences along its length beyond those noted for the ribosomal rDNA cluster, telomeres, and centromeres. The integrated map was compared with the genomic sequence of *N. crassa*, and there was substantial differences in their alignment above 9 Mb for linkage groups I and V.

KeyWords: recombination, genetic map, physical map, whole genome-shotgun sequencing, simulated annealing, MLE.

4.2 Introduction

Integration of genetic and physical maps has become a pressing problem in genomics with the steady accumulation of mapping information on a variety of genomes [7]. Genetic maps tend to have information on what genes do, but physical maps allow researchers access to the DNA sequence on genes of interest. The former are based on how genes are transmitted from generation to generation, whereas the latter require isolating and ordering distinguishable DNA fragments. Very different methodologies and experimental protocols are used to generate genetic and physical maps. As a result, the genetic and physical maps generated are not necessarily consistent. Consequently, special tools are needed to resolve rationally the discrepancies between genetic and physical maps [29] and integrate coherently their information content. In prior work we have developed maximum likelihood methods for building genetic maps ([63], [61]) based on earlier work [39]. We have developed maximum likelihood methods for building physical maps [8]. We have also investigated various combinatorial and continuous optimization algorithms and their parallel implementations to compute efficiently the maximum likelihood estimator of a physical map ([5], [6], [30]). An open methodological problem is developing maximum likelihood methods for building integrated maps of whole genomes.

Examples of physical maps vary. A *physical map* is defined to be a partial ordering of distinguishable DNA fragments [7]. The oldest example is the cytological map of the fruit fly, *Drosophila melanogaster*. A more familiar example is the DNA sequence of an entire genome. Recently there has been a shift in strategy to obtaining a genome's entire sequence [65] to what is termed the *whole genome shotgun approach* [4]. In this approach the entire genome is subdivided into many small fragments, and the resulting small fragments are then sequenced and assembled. The utility of the whole genome shotgun approach, however, is still in question [27]. The whole genome shotgun approach was employed on the model fungal system, *Neurospora crassa*, in which the 42.9 megabase (Mb) genome was sequenced to an unusual depth of > 20 genome equivalents ($> 20\times$) [24]. The question arises whether or not the resulting genomic sequence is consistent with the available genetic and physical maps (based on cosmid libraries [34]) obtained independently. The cosmid library-based physical map and genetic map provide an opportunity to evaluate the utility of the whole genome shotgun approach.

We report here for the first time a hybridization-based physical map integrated with a published genetic map for the bread mold, *N. crassa*. This integrated map has a resolution of 23 kilobases [71]. The integrated map allows an examination of whether or not repeats in the genome are organized as they are in a related fungus, *Aspergillus nidulans* [52]. Some organisms as simple as the model fungal system, *A. nidulans*, have a repeat organization similar to larger eukaryotic genomes. The question arises whether or not its relative, *N.*

crassa, has such an organization. *N. crassa* differs from *A. nidulans* in that it is intolerant to repeats and appears to have a mechanism for removing repeats from its genome [58].

We address three questions in this work. First, we address the methodological problem of how to integrate genetic and physical maps using the method of maximum likelihood. Second, we examine the efficacy of the whole genome shotgun approach in *N. crassa* against a high-resolution physical map integrated with a genetic map. The physical map is reported for the first time in this work. Lastly, we examine directly how DNA repeats are organized in a genome, in which there is experimental evidence for a process removing these repeats.

4.3 Methods and Materials

4.3.1 Probe Sampling Design

To avoid problems with repeated sequences, all cosmid probes were selected from among cosmids hybridizing to one and only one chromosome [34]. Probes with a unique assignment to a chromosome were then used to anchor the physical map to the correct chromosome and helped to achieve a more accurate reconstruction of the physical map. Cosmid probes were selected by a sampling design called *sampling without replacement* [71]. In this design each new cosmid was selected as a probe which hybridized to no previous cosmid probe. The effect of this design was to tile the chromosome with probes with no overlap between the probes. An alternate design is *sampling with replacement* in which cosmid probes are chosen randomly. As can be seen from Fig. 4.1, sampling without replacement progresses considerably faster than sampling with replacement. Progress is measured by the number of contiguous blocks of clones or *contigs* identified as probes are added into the physical mapping experiment. The size and hence time in mapping is measured by the number of probings used up to a particular time in the experiment. In the current case sampling with replacement would have taken at least twice as long (twice as many probings) to finish the physical map. The actual progress tracks the expectation under sampling without replacement, as it should, with a small departure towards the end.

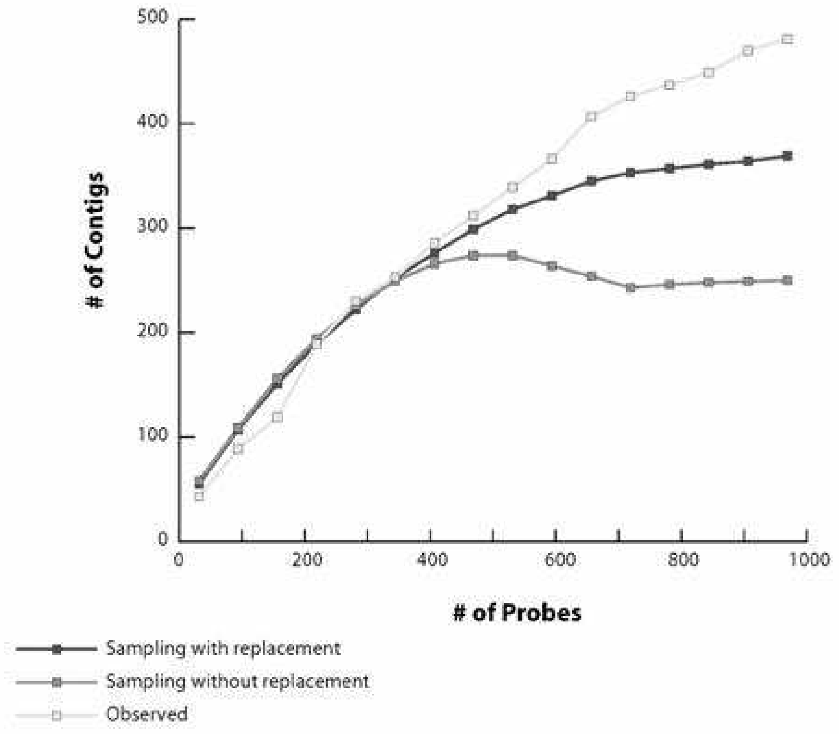


Figure 4.1: Progress Curve

4.3.2 Physical Mapping by Clone/Probe Hybridization

Each clone was assigned a binary call number (1 indicating a hbridization and 0, no hybridization) by hybridizing all clones in a chromosome-specific library [34] with a panel of more than 1000 probes as shown in Fig. 4.1. The chromosome-specific library has two parts, clones with *N. crassa* DNA inserted into the cosmid vector, pMOcosX [49], and clones with *N. crassa* DNA inserted into the cosmid vector, pLorist6Xh [34]. The average insert size was 34 kb +/- 0.7 kb [34]. Two very different cosmid cloning vectors were utilized to ensure more representative sampling of the *N. crassa* genome. Clones with inserts in pMOcosX are referred to as the pMOcosX library; clones with inserts in pLorist6Xh are referred to as the pLorist6Xh library. Clone names from these two respective libraries begin with an X or H. This letter designation is followed by a number indexing a microtitre plate, then by a letter indexing the row of a particular plate, and lastly by a number indexing the column of a particular plate. For example, H123E02 is a clone in the 123rd plate of the pLorist6Xh library in row E and column 2. Each plate has 96 clones. Libraries are available from the Fungal Genetics Stock Center at *http://www.fgsc.net*.

4.3.3 Arraying the Chromosome-Specific Libraries

The chromosome-specific cosmid libraries [34] were first double-stamped onto nylon membranes (Hybond-XL and Hybond-N+ (Amersham)) in a 6 $\times$ 6 array from 36, 96-well plates per membrane with a BioGrid high density stamping robot (BioRobotics). Inocula on membranes were allowed to grow overnight at 37^{o}C on Luria Broth agar plates containing Kanamycin (50 μg/ml) for the pLorist6Xh library or containing ampicillin (50 μg/ml) for the pMOcosX library. Membranes were treated to lyse colonies and to enrich for cosmid DNA: (1) lysing solution: 10% SDS for 5 min; (2) denaturing solution: 1.5 M NaCl, 0.5 M NaOH for 5 min; (3) neutralizing solution: 0.5 M Tris-HCl, 1.5 M NaCl, pH 7.2, for 5 min; (4) washing solution: 0.3 M NaCl, 0.3 M Na citrate, pH 7.0 (2$\times$ SSC), for 5 min. The treated membranes were then air-dried for 30 min, and DNA, cross-linked to the nylon membranes by UV radiation (3 min) with a UV cross-linker (Stratagene) or alternatively, by baking at 80^{o}C for 2 hours. The result is that all of the DNAs of the cosmids in the chromosome-specific library are arrayed on nylon membranes for hybridization with the radiolabeled DNAs of selected cosmid probes.

4.3.4 Clone/Probe Hybridization

A set of 6 membranes representing the entire chromosome-specific library in duplicate was prehybridized at 65^{o}C for 2 hours with 10-12 ml of modified hybridization buffer containing casein hydrolysate (instead of bovine serum albumin) to reduce background (non-specific) hybridization. *Gentle agitation* was implemented in a hybridization oven (Hybaid). As explained below, a probe pool was created as an equi-molar mixture of up to 7 probes, each previously assigned by clone/chromosome hybridization uniquely to one linkage group (an S-clone). Radioactive cosmid probe pools were labeled using T7 and T3 (SP6) primers flanking the insert in a pMOcosX (pLorist6Xh) clone with a random hexamer priming kit (Stratagene or High Prime(Roche)). After prehybridization, 5-20 ml of radioactive cosmid probe pool ($> 10^8$ cm) was added and hybridized overnight at 65^{o}C with gentle agitation.

Membranes were washed twice in 2$\times$ SSC, 1% SDS at 65^{o}C for 30 min with agitation. Two subsequent washes were performed in 0.5$\times$ SSC at 65^{o}C for 30 min. Washed membranes were removed, blotted dry, and electronically autoradiographed on a Packard Instant Imager (Packard, Meriden, CT). The membranes were also exposed to X-ray film (BIOMAX MR) at -80^{o}C with intensifying screens. In this way the hybridization of a probe pool to all clones in the chromosome-specific library identified whether or not each clone's DNA on the membrane shared DNA sequences in common with the probe(s).

4.3.5 Pooling Strategy of Probes

To accelerate the mapping up to 7-fold by exploiting the chromosome-specific libraries, each probe pool represented an equal mixture of up to 7 labeled chromosome-specific S-clone DNAs, and the chromosome-specific assignments of all clones in the library were then used to assign hybridization signals on autoradiographs to specific linkage groups, thereby attributing a hybridizing probe at the same time. The result was a clone/probe hybridization matrix for each linkage group.

4.3.6 Physical Mapping Strategy

A summary of the physical mapping strategy is given in Fig. 4.2. The *N. crassa* Genome Project began with separating the chromosomes of all 7 linkage groups by pulsed-filed electrophoresis using a CHEF-gel apparatus (BIO-RAD) [34]. Each chromosome was gel-isolated and radio-labeled [34] for hybridization against the pMOcosX and pLorist6Xh libraries. The result was that each of 13,882 clones was assigned to one or more chromosomes. S-clones uniquely hybridized to one chromosome. R-clones hybridized to 2 to 6 chromosomes. A-clones hybridized to all 7 chromosomes. The resulting chromosome-specific library of 13,882 clones was then used in physical mapping.

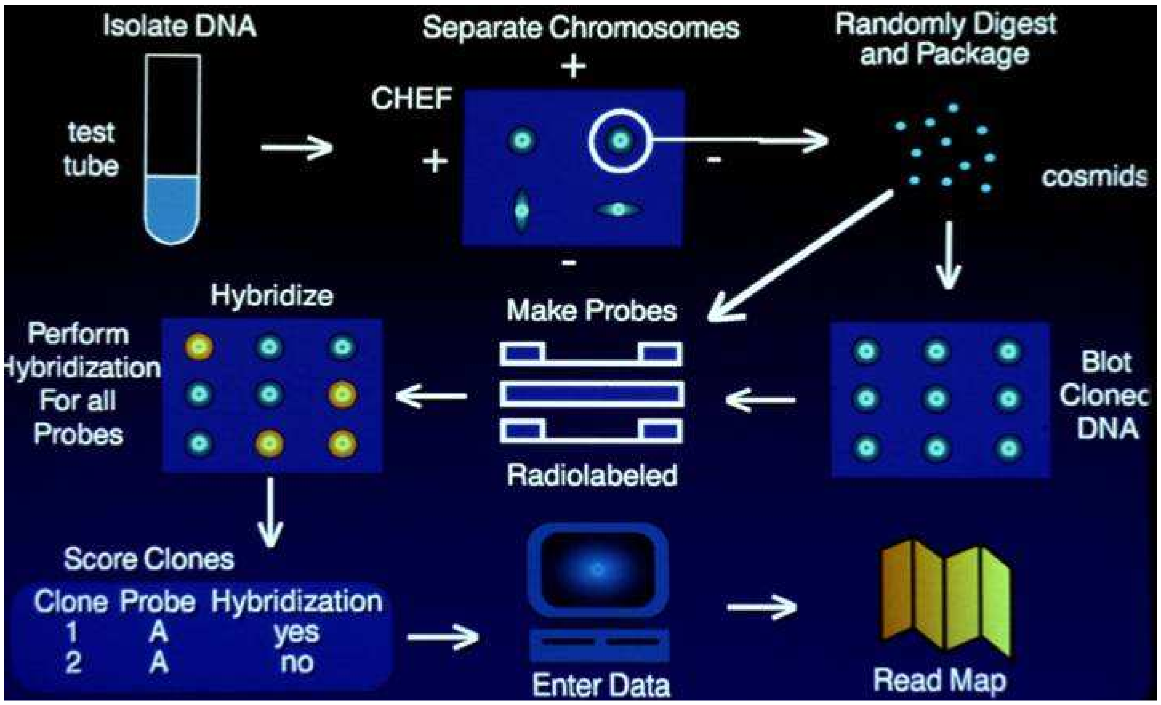


Figure 4.2: Physical Mapping Strategy

To carry out the physical mapping rapidly, the chromosome-specific library was hybridized to 312 pools of 1-7 S-clones using a sampling without replacement strategy to ensure that the probes were selected to be non-overlapping (see 4.3.1). This sampling design accelerated the rate of progress as measured by the number of contigs formed as a function of the number or probes used during the mapping experiment (Fig. 4.1). After probing the chromosome-specific library robotically arrayed on 6 nylon membranes in duplicate, the resulting positive hybridization signals were deconvoluted into linkage groups using the chromosomal assignments of each clone in the chromosome-specific library. The resulting clone/probe hybridization matrix for each linkage group (with a 1 denoting hybridization and a 0 denoting no hybridization) was uploaded to the Fungal Genome Database (FGDB) [36]. A random cost algorithm was used to compare the digital call numbers of each clone on the basis of a Hamming distance between clones to order clones in the clone/probe hybridization matrix [66]. Clones with similar digital call numbers were moved closer to each other in the hybridization matrix. An analogous process was carried out on the columns of the clone/probe hybridization matrix. Then the reordered clone/probe hybridization representing an initial physical map was manually edited to achieve a descending staircase (of positive hybridization signals) format [52].

A limitation of physical maps is their long-range continuity over 1 Mb. To overcome this problem, a single cross with 277 markers scored in progeny was used to build a genetic map [61] using published data [51]. This genetic map was aligned visually with the physical map to produce a legacy ordering for each linkage group. The legacy ordering constitutes the starting point for the likelihood analysis presented here. These initial maps and alignments were constructed without reference to the genomic sequence [24] to provide an independent test of the sequence assembly from a whole genome shotgun implemented to a depth of > 20-fold.

4.3.7 Initializing the likelihood-based search for an integrated physical and genetic map beginning with a legacy ordering

The clone/probe hybridization matrices as well as chromosome assignments of each clone, assignments of genes to clones, and a published genetic map [50] were loaded into the Fungal Genome Database (FGDB) [36]. Within the FGDB, an implementation of the random cost algorithm [66] was utilized to build physical maps for each linkage group. Finally, Dr. Arnold visually resolved the genetic and physical maps within the FGDB, to produce a legacy ordering to initialize the likelihood based reconstruction of an integrated physical and genetic map.

4.4 Likelihood Methods

The integrated log-likelihood is defined as

$$l_j = l_p + l_g \tag{4.1}$$

where, l_p denotes the physical log-likelihood and l_g denotes the genetic log-likelihood. Their expressions follow ([61]-Eq-(25), [8]-Eq-(24)) :

$$l_p = \sum_{i=1}^{k} \ln \left\{ R_i - \sum_{j=1}^{n+1} (a_{i,\pi_j} - 1)(a_{i,\pi_{j-1}} - 1) \min(Y_j, M) \right\} - C \tag{4.2}$$

where C is a constant given by

$$C = k \ln(N - M) - P \ln \frac{\rho}{(1-\rho)} - nk \ln(1-\rho) \tag{4.3}$$

and $\pi_0 = \pi_{n+1} = 0$.

$$l_g = \sum_{j=1}^{N} n_j \log \left[\sum_k \left[\frac{1}{4} \sum_{j=1}^{4} I_{\{f_i \in R_k(j,.)\}} \prod_{j=1}^{l-1} P_j \left(I_{k,j} = i_{k,j} \right) \right] \right] \tag{4.4}$$

The following sections give a very short introduction to the models used for formulating likelihood based measure on gene orders and probe orders. For detailed introduction see [61] and [8].

4.4.1 Brief Introduction to the formulation of Genetic Likelihood

We model the genotypes 1 1 , 0 0,1 0 and 0 1 (where 1 and 0 refer to paternal and maternal alleles respectively) of a genetic interval (say,S_i), as a consequence of two simultaneous independent and identical chromatid exchange events (say, S). The four possible chromatid exchanges between non-sister chromatids are pictorially represented in Figure 4.3. Let c_i denote the probability of a chromatid exchange in S between any two non-sister chromatids in the i^{th} genetic interval S_i at meiosis. Since under the *No-Chromatid-Interference* (NCI) assumption, all chromatid exchanges are equally likely, we get:

$$\begin{aligned} S &= \{0, 1, 2, 3, 4\} \\ P(i) &= \frac{c}{4}; i = 1, \cdots, 4; i \in S \\ P(0) &= 1 - c; 0 \in S \end{aligned} \tag{4.5}$$

The element 0 in S indicates the absence of an exchange event. Elements 1, 2, 3 and 4 indicate that non-sister chromatids $(1,3), (2,3), (2,4)$ and $(4,1)$ took part in the exchange

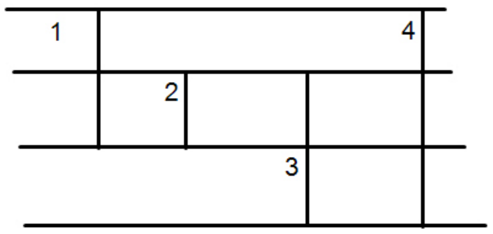


Figure 4.3: Single exchange events are signified by a vertical line, and the exchanges take place between chromatids at the ends of the vertical lines. These 4 strands are found in Prophase-I [19]. All exchanges are equally likely under our hypothesized model. Taken from [61].

process respectively. The set S_i, defined by the Cartesian product $S \times S$, enumerates all possible bivalent configurations for the i^{th} genetic interval.

The probability distribution of S_i denoting bivalent configurations between locus $A_i$and $A_{(i+1)}(i = 1, ..., l-1)$, where l=total number of loci being studied, can be derived as follows using Eqn. 4.5.

$$P(\{i,j\}) = \frac{c_i^2}{16} I_{\{i\neq 0; j\neq 0\}} + \frac{c_i(1-c_i)}{4}\{I_{\{i=0;j\neq 0\}} + I_{\{i\neq 0;j=0\}}\} + (1-c_i)^2 I_{\{i=j=0\}} \tag{4.6}$$

where, $\{i,j\} \in S_i$.

Let ϕ_k denote a unique chromatid exchange on S^l as described below:

$$\phi_k = i_1 \times i_2 \times ... \times i_{l-1} \tag{4.7}$$

where,

$$k = i_1.i_2.i_3...i_{l-1} \; ; \; i_j \in S_i \; ; \; \phi_k \in S^l = \prod_{i=1}^{l-1} S_i$$

From this point on, for the sake of brevity, we may abbreviate term *chromatid exchanges* to simply *exchanges* when referring to ϕ_k.

Let f_k denote a multi-locus genotype with l loci:

$$f_k = i_1 \times i_2 \times ... \times i_{l-1} \times i_l$$

where,

$$k = i_1.i_2.i_3...i_l \ \ ; \ \ i_j = 0, 1; \ \ \forall j = 1, \cdots, l$$

The indices $i_j = 1$ and $i_j = 0$ indicate the paternal and maternal alleles respectively. The progeny are obtained by exchanges between homogeneous parents. The observed data set can be represented as:

$$\mathcal{D} = \left\{ n_j \ \ ; \ \ \forall j = 1, \cdots, 2^l \right\}$$

where, n_j is the observed frequency of f_j.

Let us define the following functions

$$\begin{aligned} f^0(a) &= (a_1, a_2, a_3, a_4)' \\ f^1(a) &= (a_3, a_2, a_1, a_4)' \\ f^2(a) &= (a_1, a_3, a_2, a_4)' \\ f^3(a) &= (a_1, a_4, a_3, a_2)' \\ f^4(a) &= (a_4, a_2, a_3, a_1)' \\ &\text{where,} \\ a &= (a_1, a_2, a_3, a_4)' \\ a_i &= 0, 1 \ \ \forall i \end{aligned} \tag{4.8}$$

Note that Eqn. 4.8 encodes the elements of set S (See Eqn. 4.5) as mathematical functions. For example, the first function $f^0(a)$ encodes element 0, showing no chromatid exchange whereas function $f^4(a)$ encodes element 4, indicating that the first strand and the fourth strand have had an exchange.

The function $f_{ij}(a) = f_j(f_i(a))$ corresponds to the events in S_i accounting for all possible bivalent configurations. For a particular chromatid exchange ϕ_k we can generate a model tetrad at meiosis using the function f_{ij}. The matrix R_k of size $4 \times l$ defines the simulated tetrad as follows [57]:

$$R_k = \left(R_0 R_1 \cdots R_{(l-1)} \right) \tag{4.9}$$

where,

$$R_0 = (1100)' \ \ ; \ \ R_i = f_{jk}(R_{i-1}) \ \forall i = 1, \cdots, l-1$$

and the i^{th} genetic interval S_i contains the observed chromatid exchange $\{j,k\}$. In other words, R_k has 4 rows which correspond to the 4 gametes in a tetrad during meiosis if the chromatid exchange ϕ_k had occurred according to our model.

The conditional distribution of f_i for a given ϕ_k is

$$P(f_i|\phi_k) = \frac{1}{4}\sum_{j=1}^{4} I_{f_i \in R_{k(j,.)}} \tag{4.10}$$

where, $R_k(j,.)$is the j^{th} row of R_k.

The marginal density of a single spore f_i is given by

$$\begin{aligned} P(f_i) &= \sum_k P(f_i|\phi_k) \times P_k \\ &= C \times P \end{aligned} \tag{4.11}$$

where, C is the conditional probability matrix given by :

$$\left.\begin{aligned} C &= ((c_{ki})) \\ c_{ki} &= P(f_i|\phi_k) \quad \text{(from equation (4.10))} \end{aligned}\right\} \tag{4.12}$$

and P is given by,

$$\begin{aligned} P &= (P_k;\ \ \forall k)' \\ P_k &= P(\phi_k) \\ &= \prod_{j=1}^{l-1} P\left(I_j = i_{k,j}\right) \end{aligned} \tag{4.13}$$

where, $i_{k,j} \in S_j$ and the probability distribution $P\left(I_j = i_{k,j}\right)$ is as defined in equation (4.6).

Let $\Theta = (c_1, c_2, ..., c_{l-1})'$ denote the unknown parameter vector in the model. The log-likelihood of the counts $\mathbf{f} = (f_i, i = 1, \cdots, 2^{l-1})'$, viewed as a function of Θ, is given by :

$$l_g(\Theta|D) = \sum_{i=1}^{N} n_i \log\left[\sum_k \left[\frac{1}{4}\sum_{j=1}^{4} I_{\{f_i \in R_k(j,.)\}} \prod_{j=1}^{l-1} P_j\left(I_{k,j} = i_{k,j}\right)\right]\right] \tag{4.14}$$

Note that the log-likelihood in Eqn. 4.14 is distinct from the one in Zhao *et.al* [73]. Zhao and Speed have formulated a log-likelihood function similar to the one in Eqn. 4.14 for ordered tetrads in the appendix of [75]. However, the log-likelihood in Eqn. 4.14 does not hypothesize an exchange process. Both, Lander and Green [39] and Zhao and Speed [75] have used the EM algorithm in the context of genetic map reconstruction.

The following two theorems maximize the log-likelihood in Eqn. 4.14 using a set of recurrence relations obtained via the Expectation-Maximization (EM) algorithm [20]. The proofs are not given in the interest of brevity, but the development of Eqn. 4.15 is described elsewhere [61].

Theorem 4.1. *The EM-iterative equations are given below.*

$$\begin{aligned}\Theta^{(h+1)} &= \left(c_m^{(h+1)} \;\; \forall m = 1, \cdots, l-1\right)' \\ \textit{where } c_m^{(h+1)} &= \left(\frac{2N_{2,m} + N_{1,m}}{2N_m}\right)^{(h)}\end{aligned} \tag{4.15}$$

where

$$N_m = \sum_k n_k^{(h)} = N_{0,m} + N_{1,m} + N_{2,m}$$

$$N_{0,m} = \sum_{k \big| i_{k,m}=(0,0)} n_k^{(h)}$$

$$N_{1,m} = \sum_{\substack{k \big| i_{k,m}=(i_1,i_2) \\ i_1=0 \text{ (Strict)OR } i_2=0}} n_k^{(h)}$$

$$N_{2,m} = \sum_{\substack{k \big| i_{k,m}=(i_1,i_2) \\ i_1\neq 0 \text{ AND } i_2\neq 0}} n_k^{(h)}$$

$$n_k^{(h)} = \sum_j n_j \pi_{k|j}(\Theta^{(h)})$$

$$\pi_{k|j}^{(h)} = P(x_{kj} = 1 | f_j) = \frac{\pi_{j|k}^{(h)} \times \pi_k^{(h)}}{p_j^{(h)}}$$

$$p_j^{(h)} = \sum_k \pi_{j|k}^{(h)} \times \pi_k^{(h)}$$

$$\pi_{j|k}^{(h)} = c_{ki} \textit{ in Eqn. 4.12 for the } h^{th} \textit{iteration}$$

$$\pi_k^{(h)} = P_k \textit{ in Eqn. 4.13 for the } h^{th} \textit{iteration}$$

Note that, $i_{k,m}$ denotes an event in S_m for the exchange ϕ_k.

The following theorem provides an initialization of c for the recurrence relations in Eqn. 4.15.

Theorem 4.2. *Let* $\mathbf{f} = (f_1, f_2, f_3, f_4)'$ *be the observed frequency vector corresponding to all possible meiotic products for parental genes M and O for two markers. The genotype vector for* $\mathbf{f}$ *is (MM MO OM OO)'. The maximum likelihood estimator [54] of the exchange probability* c *under the model represented by Eqn. 4.5 is unique and is given as follows:*

1. *If* $f_1 + f_4 < f_2 + f_3$ *then* $c_{mle} = 1$
2. *If* $f_1 + f_4 \geq f_2 + f_3$ *then* c_{mle} *is given by the unique solution (in the interval* $[0, 1]$ *) of the following equation:*

$$f(c) = c^2 - 2c + D = 0 \tag{4.16}$$

where,

$$D = \frac{2(f_2 + f_3)}{N} \; ; \; N = \sum_{i=1}^{4} f_i$$

This theorem is used to obtain the starting values of c_m *for the EM-iterative equations in Theorem (4.1).*

4.4.2 Brief Introduction to the formulation of Physical Likelihood

The probe ordering problem can be formally stated as follows. Given a set $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ of n probes and a set $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ of k clones generated using the sampling-without-replacement protocol described earlier, and the $k \times n$ clone-probe hybridization matrix H containing both false positives and false negatives with predefined probabilities, reconstruct the correct ordering $\Pi = (\pi_1, \pi_2, \ldots, \pi_n)$ of the probes and also the correct spacing $Y = (Y_1, Y_2, \ldots, Y_n)$ between the probes. The ordering Π is a permutation of $(1, \ldots, n)$ that gives the labels (indices) of the probes in left-to-right order across the chromosome. In the inter-probe spacing vector Y, Y_1 denotes the space between the left end of the first probe P_{π_1} and the left end of the chromosome, and Y_i the spacing between the right end of probe $P_{\pi_{i-1}}$ and and the left end of probe P_{π_i} (where $2 \leq i \leq n$). The spacing between the right end of probe P_{π_n} and the right end of the chromosome is given by $Y_{n+1} = N - nM - \sum_{i=1}^{n} Y_i$ where N is length of the chromosome and M is the length of each clone/probe. Note that our protocol requires that all probes and clones be of the same length.

The mathematical notation used in the formulation of the maximum likelihood estimator is given below:

N : Length of the chromosome,

M : Length of a clone/probe,

n : Number of probes,

k : Number of clones,

ρ : Probability of false positive,

η : Probability of false negative,

$H = ((h_{i,j}))_{1 \le i \le k,\, 1 \le j \le n}$: clone-probe hybridization matrix,

where

$$h_{i,j} = \begin{cases} 1 & \text{if clone } C_i \text{ hybridizes with probe } P_j \\ 0 & \text{otherwise,} \end{cases}$$

H_i : ith row of the hybridization matrix,

$\Pi = (\pi_1, \ldots, \pi_n)$: permutation of $\{1, 2, \ldots, n\}$ which denotes the probe labels in the ordering when scanned from left to right along the chromosome,

$p_i = \sum_{j=1}^{n} h_{i,j}$: number of 1's in H_i,

$P = \sum_{i=1}^{k} p_i$: total number of 1's in H, $Y = (Y_1, Y_2, \ldots, Y_n)$: vector of inter-clone spacings where Y_i is the spacing between the right end of $P_{\pi_{i-1}}$ and the left end of P_{π_i} $(2 \le i \le n)$, and Y_1 is the spacing between the left end of P_{π_1} and the left end of the chromosome, and $\mathcal{F} \subseteq \mathcal{R}^n$: set of feasible interprobe spacings $Y = \{Y_1, \ldots, Y_n\}$ such that $Y_i \ge 0$, $1 \le i \le n$ and $N - nM - \sum_{i=1}^{n} Y_i \ge 0$.

The Model: Given a vector of inter-probe spacings $Y = (Y_1, \ldots, Y_n)$, there are 2^{n+1} possible cases to consider depending on whether $0 \le Y_i \le M$ or $Y_i > M$ where $0 \le i \le n+1$. Without loss of generality, we present the maximum likelihood model for $n = 3$ and illustrate 3 of the $2^4 = 16$ possible cases.

Case 1: $0 \leq Y_1 \leq M$, $0 \leq Y_2 \leq M$, $0 \leq Y_3 \leq M$, and $0 \leq Y_4 \leq M$ as depicted in Figure 4.4.

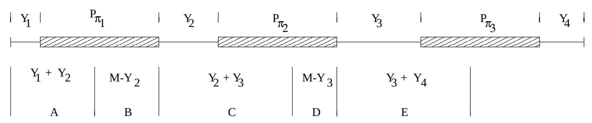


Figure 4.4: Interprobe spacings: Case 1

Case 2: $0 \leq Y_1 \leq M$, $0 \leq Y_2 \leq M$, $0 \leq Y_3 \leq M$, and $Y_4 > M$ as depicted in Figure 4.5.

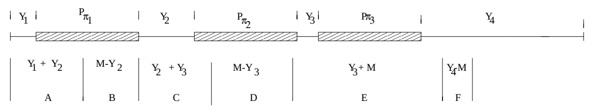


Figure 4.5: Interprobe spacings: Case 2

Case 3: $Y_1 > M$, $0 \leq Y_2 \leq M$, $0 \leq Y_3 \leq M$, and $0 \leq Y_4 \leq M$ as depicted in Figure 4.6.

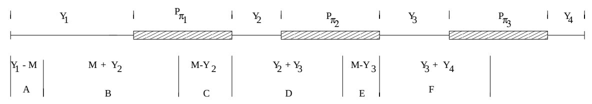


Figure 4.6: Interprobe spacings: Case 3

Type 1: *Both* region between probe P_{π_j} and $P_{\pi_{j+1}}$, for $j = 1, \ldots, n-1$. An intervening clone hybridizes to both probes if its left end falls in this region.

Type 2: *Only* region of probe P_{π_j}, for $j = 1, \ldots, n$. A clone will hybridize to P_{π_j} only of its left end falls in this region.

Type 3: *None* region after probe P_{π_j}, for $j = 0, \ldots, n$. A clone will hybridize to no probe if its left end falls in this region. Here probe P_{π_0} is referred to as the beginning of the chromosome.

It can be shown that for $j = 1, \ldots, n-1$

$$\begin{aligned} \begin{array}{l} \text{Length of the } \textit{Both} \text{ region} \\ \text{between probes } P_{\pi_j} \text{ and } P_{\pi_{j+1}} \end{array} &= \begin{cases} 0 & \text{if } Y_{j+1} > M \\ M - Y_{j+1} & \text{if } Y_{j+1} \leq M \end{cases} \\ &= M - \min(Y_{j+1}, M), \qquad (4.17) \end{aligned}$$

and for $j = 1, \ldots, n$

$$\begin{array}{c}\text{Length of the } \textit{Only} \\ \text{region of probe } P_{\pi_j}\end{array} = \begin{cases} Y_j + Y_{j+1} & \text{if } Y_j \leq M \text{ and } Y_{j+1} \leq M \\ M + Y_j & \text{if } Y_j > M \text{ and } Y_{j+1} \leq M \\ Y_j + M & \text{if } Y_j \leq M \text{ and } Y_{j+1} > M \\ 2M & \text{if } Y_j > M \text{ and } Y_{j+1} > M \end{cases}$$
$$= \min(Y_j, M) + \min(Y_{j+1}, M), \tag{4.18}$$

and for $j = 0, \ldots, n$

$$\begin{array}{c}\text{Length of the } \textit{None} \\ \text{region after probe } P_{\pi_j}\end{array} = \begin{cases} Y_{j+1} - M & \text{if } Y_{j+1} > M \\ 0 & \text{if } Y_{j+1} \leq M \end{cases}$$
$$= Y_{j+1} - \min(Y_{j+1}, M). \tag{4.19}$$

We assume that the left ends of the clones are uniformly distributed over the interval $[0, N - M]$ i.e., the probes are uniformly distributed across the length of the chromosome. Therefore it can be shown that for $j = 1, \ldots, n-1$, the probability P_{Both} that a randomly chosen clone will fall in the *Both* region of probes P_{π_j} and $P_{\pi_{j+1}}$ is given by

$$P_{Both} = \frac{M - \min(Y_{j+1}, M)}{N - M}; \tag{4.20}$$

for $j = 1, \ldots, n$ the probability P_{Only} that a randomly chosen clone will fall in the *Only* region of probe P_{π_j} is given by

$$P_{Only} = \frac{\min(Y_j, M) + \min(Y_{j+1}, M)}{N - M}, \tag{4.21}$$

and for $j = 0, \ldots, n$ the probability P_{None} that a randomly chosen clone will fall in the *None* region after probe P_{π_j} is given by

$$P_{None} = \frac{Y_{j+1} - \min(Y_{j+1}, M)}{N - M}. \tag{4.22}$$

The conditional probability of observing a clonal signature H_i (i.e., the ith row in the hybridization matrix H), given a probe ordering Π and an inter-probe spacing vector Y, is given by

$$\begin{aligned}P(H_i \mid \Pi, Y) &= \sum_{j=1}^{n} P(H_i \mid \Pi, Y, O_{i,j}) P(O_{i,j} \mid \Pi, Y) + \\ &\sum_{j=1}^{n-1} P(H_i \mid \Pi, Y, B_{i,j}) P(B_{i,j} \mid \Pi, Y) + \\ &\sum_{j=0}^{n} P(H_i \mid \Pi, Y, N_{i,j}) P(N_{i,j} \mid \Pi, Y) \qquad (4.23)\end{aligned}$$

where

$O_{i,j}$: the event that the clone i will fall in the *Only* region of probe P_{π_j}.

$B_{i,j}$: the event that the clone i will fall in the *Both* region of probes P_{π_j} and probe $P_{\pi_{j+1}}$.

$N_{i,j}$: the event that the clone i will fall in the *None* region after probe P_{π_j}.

Assuming that the false positive and false negative errors at different positions along the clonal signature H_i are independent of each other the following can be shown

$$P(H_i \mid \Pi, Y, O_{i,j}) = (1-\eta)^{h_{i,\pi_j}} \cdot \eta^{(1-h_{i,\pi_j})} \cdot \rho^{(p_i - h_{i,\pi_j})} \cdot (1-\rho)^{(n-1)-(p_i-h_{i,\pi_j})} \qquad (4.24)$$

$$\begin{aligned}P(H_i \mid \Pi, Y, B_{i,j}) &= (1-\eta)^{(h_{i,\pi_j}+h_{i,\pi_{j+1}})} \cdot \eta^{(2-h_{i,\pi_j}-h_{i,\pi_{j+1}})} \cdot \rho^{(p_i-h_{i,\pi_j}-h_{i,\pi_{j+1}})} \\ &\quad \cdot (1-\rho)^{(n-2)-(p_i-h_{i,\pi_j}-h_{i,\pi_{j+1}})} \qquad (4.25)\end{aligned}$$

$$P(H_i \mid \Pi, Y, N_{i,j}) = \rho^{p_i} \cdot (1-\rho)^{(n-p_i)}. \qquad (4.26)$$

From the previous equations,

$$\begin{aligned}P(H_i \mid \Pi, Y) &= \sum_{j=1}^{n} \Big[(1-\eta)^{h_{i,\pi_j}} \cdot \eta^{(1-h_{i,\pi_j})} \cdot \rho^{(p_i-h_{i,\pi_j})} \cdot (1-\rho)^{(n-1)-(p_i-h_{i,\pi_j})} \\ &\quad \cdot \frac{\min(Y_j, M) + \min(Y_{j+1}, M)}{N-M}\Big] \\ &\quad + \sum_{j=1}^{n-1} \Big[(1-\eta)^{(h_{i,\pi_j}+h_{i,\pi_{j+1}})} \cdot \eta^{(2-h_{i,\pi_j}-h_{i,\pi_{j+1}})} \cdot \rho^{(p_i-h_{i,\pi_j}-h_{i,\pi_{j+1}})} \\ &\quad \cdot \frac{M - \min(Y_{j+1}, M)}{N-M}\Big] \\ &\quad + \sum_{j=0}^{n} \Big[\rho^{p_i} \cdot (1-\rho)^{(n-p_i)} \cdot \frac{Y_{j+1} - \min(Y_{j+1}, M)}{N-M}\Big] \qquad (4.27)\end{aligned}$$

We assume that the clones $\in \mathcal{C}$ are independently distributed along the chromosome i.e., each row of H is independent of the other rows. Hence

$$P(H \mid \Pi, Y) = \prod_{i=1}^{k} P(H_i \mid \Pi, Y). \tag{4.28}$$

From equations (4.27) and (4.28),

$$P(H \mid \Pi, Y) = \prod_{i=1}^{k} C_i \left\{ R_i - \sum_{j=1}^{n+1} (a_{i,\pi_j} - 1)(a_{i,\pi_{j-1}} - 1) min(Y_j, M) \right\} \tag{4.29}$$

where

$$a_{i,j} = \begin{cases} \frac{\eta}{(1-\rho)} & \text{if } h_{i,j} = 0 \text{ and } j = 1, \dots, n \\ \frac{(1-\eta)}{\rho} & \text{if } h_{i,j} = 1 \text{ and } j = 1, \dots, n \\ 0 & \text{otherwise,} \end{cases} \tag{4.30}$$

$$C_i = \frac{\rho^{p_i}(1-\rho)^{(n-p_i)}}{N - M}, \tag{4.31}$$

and

$$R_i = N - nM + M \sum_{j=1}^{(n-1)} a_{i,\pi_j} a_{i,\pi_{j+1}}. \tag{4.32}$$

Hence the log-likelihood function is given by

$$l_p = \sum_{i=1}^{k} \ln \left\{ R_i - \sum_{j=1}^{n+1} (a_{i,\pi_j} - 1)(a_{i,\pi_{j-1}} - 1) \min(Y_j, M) \right\} - C \tag{4.33}$$

where C is a constant given by

$$C = k \ln(N - M) - P \ln \frac{\rho}{(1-\rho)} - nk \ln(1-\rho) \tag{4.34}$$

and $\pi_0 = \pi_{n+1} = 0$.

4.5 Finding Best Order Using Simulated Annealing

Several combinatorial optimization problems can be tackled using simulated annealing (SA) as a stochastic optimization technique [35]. We use SA to determine the best order of genetic markers by sampling stochastically from the space of all possible gene orders guided by l_g.

Simulated annealing has been used previously [17], to reconstruct chromosomes based on binary scoring of DNA fragments and a Hamming distance-based objective function. In our case, for the genetic map, the objective function is the genetic likelihood l_g (see Section 4.4) of a particular order of genes. This objective function is computed upon convergence of the EM algorithm ([45], [67]). Thus, in our case a single computation of the objective function for the genetic map for a single order of markers is quite expensive.

For the physical map, the objective function is the physical likelihood l_p (see Eqn 4.2) of a particular order of probes. The computation of this objective function involves finding an optimal spacing among the probes using gradient descent search [8]. For the integrated physical and genetic map we locate the genes on the probes (using sequence information,complementation and hybridization) and then for a an order of probes derive and order of genes that are tagged while keeping the un-tagged genes as they are. The objective function for the integrated map is the integrated likelihood l_j (see Eqn 4.1). The SA provides a stochastic sampling technique to search for a good order as follows:

1. Choose a random order of probes, Π, and calculate $f(\Pi)$.

2. Choose a random segment within the ordering Π.

3. Perform a segment reversal and call the new ordering Π'.

4. Compute $f(\Pi')$.

5. If $f(\Pi')$ is less than $f(\Pi)$, then retain the new order. However, if $f(\Pi')$ is larger than $f(\Pi)$, then generate a random number between 0 and 1. If this random number is less than $E(-(f(\Pi') - f(\Pi))/T$, then retain the new order. Here T is the "temperature" of the annealing schedule.

6. Proceed to anneal the temperature in multiplicative steps, i.e., decrease T by a factor F at each SA iteration. Hold each value of T constant until D re-orderings have been attempted or S successful re-orderings have resulted, whichever comes first. If the number of successes equals zero for a given step, the process is complete; otherwise go to step 2.

For each order in SA, the likelihood l is computed, and the order with the maximum value of the likelihood is selected as the desired map. The values of the parameters used in the search of the best order are $T = 50.0$,$F = 0.95$,$D = 500$ and $S = 2000$.

4.6 Results

4.6.1 Quality of Integrated Physical Map

For the 42.9 Mb *N. crassa* genome a chromosome-specific library of 13,882 clones or 11 genome equivalents was constructed with an average insert size of 34 kb [34]. A total of 10,356 clones were uniquely assigned to linkage groups and called S-clones; 3,419 clones were assigned to 2-6 linkage groups (R-clones); and 107 clones were assigned to all 7 linkage groups (A-clones) [34]. A total of 1506 probings with principally non-overlapping S-clones (see 4.3.1) from this 13,882 clone chromosome-specific library [34] was used to create an integrated physical and genetic map or *integrated physical map* for short. The 882 cosmid probes produced a total of 22773 positive hybridization signals, and each probe on average hybridized to 25 clones. The clone/probe hybridization H provided the linking information to produce the physical map, which was then integrated with a genetic map [51]. The physical map consists of 176 contigs across 7 linkage groups (Fig. 4.8). The average number of clones per contig is 8, and the average size of a contig is 201 kb.

The quality of the resulting integrated physical map can be assessed in three ways, by its completeness, by its resolution, and by its consistency with itself, the genetic map, and sequenced-based physical map [24]. Sizes of the chromosomes are known [48] and can be compared with the chromosome sizes estimated from the physical map. The resolution is determined by the average spacing between markers along the physical map. The consistency involves comparing the hybridization-based physical map with itself and the genetic map and genomic sequence.

4.6.2 Completeness

The completeness of the physical map can be assessed in several ways. One approach is by the number of contigs in the final map. The most complete physical map of a eukaryote (organisms with nuclei) is likely to remain the 12 Mb genomic sequence of the fungus, *Saccharomyces cerevisiae* [26], where there are 16 contigs corresponding to its 16 linkage groups. In contrast, the genomic sequence of the fungus, *N. crassa*, consists of 958 contigs assembled into what are called scaffolds (163 in number). The integrated physical map reported here consists of 175 contigs of overlapping cosmids. As clone/probe hybridizations were performed, the completeness of the project was monitored sequentially under sampling without replacement as shown in Fig. 4.1 to examine the progress of the project.

Completeness can also be assessed by taking the estimated size of each linkage group derived directly from its physical map. As an example, in Fig. 4.4 and Fig. 4.5, linkage group I's physical map includes the ordering of 153 tiling cosmid probes, each with an insert of 34 kb on average, and the maximum likelihood estimates of the spacings $Y_1, Y_2, , Y_{153}$ between tiling probes. The estimated lengths of tiling probes and spacings can be summed to yield an estimated size of each linkage group's chromosome in Table 4.2, i.e., the physical size in kb. The lowest coverage is for linkage group I (85.7 %). The coverage for the remaining linkage groups is over 97%. The estimated completeness for the entire genome from Table 4.2 is 96%. An independent assessment of completeness was performed using 344 genes assigned to clones in the pMOcosX library derived from the Fungal Genetics Stock Center (*http://www.fgsc.net*) and sought in the physical map, and 318 of these genes were located on the physical map, suggesting a completeness of 92%. In contrast to the genomic sequence [24], the ribosomal rDNA cluster [38] is found on the physical map, although it lacks telomere ends [55] and probably portions of the centromeres as discussed previously [34]. A precise estimate of the sizes of gaps $Y_i > 34$ kb constituting contig boundaries is given in Fig. 4.9 along with gap locations as a byproduct of the maximum likelihood method used for map reconstruction. Unlike most published physical maps, this one tells us what we know and how much is not known.

4.6.3 Resolution

The *resolution* of a physical map is determined by the number of markers and the spacings between them on average. This quantity can also be derived from the maximum likelihood estimation of the physical map (see Section 4.4) as well. The *markers* on the physical map are the ends of clones in the chromosome-specific library on the physical map. The *resolution* is the average spacing between ends. As shown in Fig. 4.9, the estimated spacing alternates between the insert size, 34 kb, and an estimated spacing, Y_i. The resolution is 23 kb. This is considerably higher than that of the genetic map with its 252 markers [50] used to assess the long-range accuracy of the genomic sequence [24] or the restriction fragment length polymorphism (RFLP) genetic map with its 277 markers used here [51]. The genetic maps place a marker at 150 kb intervals throughout the genome, if they were distributed at random in the genome. The resolution of the physical map provides a much stronger test of the long-range accuracy of the genomic sequence.

4.6.4 Consistency

Self-consistency can be assessed by examining the clone/probe hybridization matrix for each linkage group after maximum likelihood estimation (see Section 4.4) at http://gene.genetics.uga.edu or [2]. Once the map is complete, the physical map can be represented by a descending staircase of positive hybridization signals in a recordered clone/probe hybridization matrix stretching from one end of the reconstructed chromosome to the other end (See, for example, [52] or [2]). If there were no errors in the data, under sampling without replacement each clone should only hybridize to 0, 1, or 2 probes (Fig. 4.4) with hybrdization signals falling near the diagonal of the reordered clone/probe hybridization matrix. Signals in the clone/probe hybridization matrix not satisfying this property are explained as false positives or false negatives with rates of 0.02 in the maximum likelihood procedure [8]. The repeatability of hybridization also puts an upper bound on the sum of the false positive and false negative rate of about 5% [34]. Entries off the diagonal of the descending staircase in the reordered clone/probe hybridization matrix could also be explained by the presence of repeated sequences in the genome or be hybridization to chimeric clones with inserts derived from 2 or more regions of the genome. Given that each off-diagonal positive entry is the result of two independent hybridizations on double-stamped membranes, the latter explanations of repetitive sequences or chimeric clones may be more likely (but harder to deal with in constructing a likelihood). If these off-diagonal positive entries were all false positives, the estimated rate would be 0.3% or less here and 3% in [2]. In the likelihood-based construction of the physical maps we thus used false positive (ρ) and false negative (η) rates of 0.2%. Consistency of the physical and genetic maps with the sequence is taken up in later sections.

4.6.5 Integration of physical and genetic maps

The major methodological problem solved here is integrating a physical and genetic map with the method of maximum likelihood, as illustrated here with the building of an integrated map of *N. crassa.* To achieve this goal first a model and methodology for estimating a physical map by the method of maximum likelihood was developed for this genome project ([8]; [33]). The limitation of this solution is that long-range continuity of the physical map was not insured. False joins do inevitably enter into the physical map to result in contigs that are overly long and inappropriately joined [66]. To overcome this problem, we developed a separate methodology for building a genetic map by the method of maximum likelihood ([63], [61]) with the goal of using the genetic map to give accurate long-range continuity to the physical map. The unusual challenge here is that experiments can be done in fungi to examine directly the gametic products of a cross through tetrad analysis, giving much more information about recombination than is normally available in building a genetic maps for other organisms [19]. Secondly, by developing a model for tetrad analysis and the associated maximum likelihood methods, we did not have to rely on a composite genetic map [50] built from many crosses involving parents of many different genetic backgrounds to validate the physical map, as done with the genomic sequence [24]. Instead we could rely on a genetic map produced by a single cross of two parents, thus permitting the quantitative estimation of recombination distances between 277 markers as well as the inference of their order for building a physical map with long-range continuity. With both methodologies in hand we could then construct a joint log-likelihood function l_j in Eqn 4.1 to reconstruct an integrated map here. The unusual challenge of this estimation problem is that one set of parameters, the orderings of markers on each map, is discrete, while the distances between markers on each map are real-valued.

The two log-likelihoods, l_g and l_p, in Eqn. 4.1 share some parameters and have their own unique parameters. The unique parameters to the genetic log-likelihood l_g are the exchange probabilities c_i between markers giving rise to the genetic map distances between markers and their orderings on the genetic map and not assigned to the physical map. The unique parameters to the physical mapping log-likelihood l_p are the cosmid probe spacings y_i and ordering of cosmids not linked to the genetic map through genes which they carry. The shared parameter is the ordering of genetic markers assigned to both maps. The ordering of shared markers creates a tension between the two likelihood functions, whose resolution is the integrated maps of each linkage group as shown in Fig. 4.7.

The coherence in map alignments can be examined at the level of genes shared between the maps and at the finer level of cosmid probes linked between maps by the genes carried. The coherence of the aligned maps is graphically displayed in Fig. 4.7 at the level of genes. The ordering of shared markers are displayed on the genetic, integrated, and physical maps by connecting common markers on different maps. If the order of shared markers differ, the lines cross. The largest linkage groups (I and V) have the most incoherence as measured by the number of line crossings for each linkage group in Table 4.3, a phenomenon seen for the other physical map, the genomic sequence [61]. The integrated map is identical to the physical map at the level of genes (but not at the cosmid probe level, results not shown). The reason for this is that the search for the integrated map was initialized with a legacy ordering that was visually aligned to the genetic map (See Materials and Methods). Apparently the legacy ordering was a very good initialization. This phenomenon also occurred for the genetic maps [61]: the maximum likelihood ordering was largely coincident with the heuristically constructed genetic maps published [51]. Use of the legacy ordering for initialization did improve the value of l_j in Eqn 4.1 (Table 4.3) vs. other initializations tried, such as the original order in which probes were collected in Fig. 1 (see Table 4.1); however, a l_j value could be maximized in a few minutes on a laptop computer that was competitive with the legacy ordering without investing several months of time required in producing the legacy ordering (see Table 4.3) as described under Materials and Methods.

Table 4.1: Likelihood

Linkage Group	l_g	l_j	l_p	l_s
1	−42.011	−25659.4155	−25626.7398	−334.5498
2	−63.9341	−17649.6358	−17600.3810	−180.1985
3	−101.1718	−14663.7926	−15224.1471	−82.7432
4	−86.1272	−18660.5664	−18588.0228	−237.2802
5	−4.7988	−15812.1735	−35379.9591	−299.7167
6	−116.3973	−15812.1735	−15763.7036	−178.5936
7	−60.3366	−13428.6187	−13418.7729	−141.5237

In linkage group I the joint order does differ from that on the genetic map. The main inconsistency appears to be a block of genes, *nuo12.3-al-2-vma-11*, that are placed on different arms of the chromosome for linkage group I for the integrated and genetic maps. The order of this block on the RFLP map [51] is supported by the composite genetic map [50]. Most geneticists would conclude that the integrated map is in error.

In linkage group IV the genetic and physical maps have a better alignment with fewer line crossings (Fig. 3). There is still one marker *cre-1* that is assigned to a very different location on the genetic map, and again the position assigned by the RFLP map [51] is concordant with the assignment of the composite genetic map [50]. The alignments of maps at the gene level as well as compressed and full uncompressed integrated maps [52] are reported at *http://gene.genetics.uga.edu*. With these integrated maps several interesting biological questions can be addressed.

4.6.6 Centromere effect

In addition to centromeres allowing chromosomes to move in mitosis and meiosis (i.e., processes of cell division), they have an effect on recombination known as the centromere effect in a variety of organisms [14]. The centromere effect is an hypothesis in which recombination is reduced near centromeres, and hence the physical size of a map unit (Mu) on the genetic map (1 Mu is 1% percent recombination on the genetic map) increases near the centromere. The centromere of linkage group VII has been partially characterized and evidence, presented for a centromere affect in *N. crassa* [12]. The question is whether or not all centromeres in *N. crassa* display this phenomenon. This question can be addressed by virtue of the centromere markers on the integrated map (see Fig. 3). In the case of linkage group IV, the gene *aod-1* is linked to the centromere [51].

Each marker's distance in kb divided by its distance in map units to its nearest neighbor distal to the centromere on the integrated maps at the gene level was plotted against physical distance from the centromere in Fig. 5.1.A using the physical map. As markers recede from the centromere, the physical size of a map unit appears to decline with distance from the centromere. The centromere effect as an hypothesis is supported. In addition it can be seen that there is an order of magnitude variation in physical size of a map unit ranging from 24 kb to 871 kb per Mu. This is typical of the variation in the expansion and contraction of the genetic map relative to a physical map [43] that is seen in a variety of organisms. The explanation for this variation in the physical size of a map unit in *N. crassa* apparently is the presence of genes that locally affect recombination rate ([9]; [10]). A genetic map is thus analogous to a rubber band that is stretched or allowed to relax at different places next to the analog of a ruler, a physical map.

4.6.7 Direct Empirical determination of the mapping function relating recombination fraction to physical distance along the chromosome

There is a considerable body of literature attempting to establish a functional relation between recombination distance on the genetic map to physical distance in kb or to at least the underlying recombination process [73], and this literature dates back to the earliest effort of [28] to calculate this relationship. This relationship is termed a mapping function derived from a model of recombination [28]. Barratt *et al.* [3] in the same spirit have modeled the recombination process in *N. crassa* and produced a mapping function as well relevant for tetrad analysis. In all of these studies the evidence in support of a particular mapping function is indirect.

Since all pairs of neighboring markers on the integrated map in Fig. 5.1 have both an estimated recombination fraction [61] in one genetic background as well as a physical distance, we can empirically determine the mapping function. In Fig. 5.1B the recombination distance $r_i = c_i(1 - c_i/2)$ in map units between neighboring markers is plotted as a function of physical distance (see [61], for a derivation of the quadratic relation between the recombination fraction r_i and the underlying exchange probability between nonsister chromatids c_i). This empirically derived mapping function compares favorably with both the simplest Haldane mapping function, but this mapping function obviously has its limitations. It does not take into account other phenomena, such as the centromere effect, or loci that affect the recombination rate locally ([9], [10]).

4.6.8 Evaluation of the whole genome shotgun sequencing strategy

The National Science Foundation (NSF) and the National Institutes of Health (NIH) are now investing hundreds of millions of dollars in this approach with little critical evaluation of its limitations [27]. The availability of an integrated physical and genetic map at 23 kb resolution in *N. crassa* provides an unusual opportunity to evaluate critically the whole genome shotgun approach in an instance most favorable to this approach. The whole genome shotgun in *N. crassa* was carried out to a depth of $>$ 20-fold, which is much deeper than in most other systems.

As previously noted [61], there were differences in gene orderings between the genetic map and genomic sequence for linkage groups I, II, and V, although the maximum-likelihood-based genetic map actually had a lower log-likelihood l_g than that of the sequence map for linkage group II. What remained as unsatisfactory was the alignment of the two largest linkage groups (I and V in Table 4.2) at the gene level with the genomic sequence; there was substantial incoherence with the maximum likelihood estimate of the RFLP map used here [61]. As a consequence the alignment of the genetic and integrated maps was compared further with that of the genomic sequence at the cosmid probe level as a stronger test of the whole genome shotgun approach. As seen in Fig. 4.8, the genomic sequence had substantial number of line crossings in Table 4.3 (and visually in Fig. 4.8) for linkage groups I and V.

One limitation then of the genomic sequence is in long-range continuity beyond 9 Mb. There is more coherence between the integrated map and genomic sequence with the 5 remaining smaller chromosomes over a range of sizes up to 5.7 Mb (Table 4.2).

Table 4.2: Estimated Sizes of Chromosomes of *Neurospora crassa*

Linkage Group	Size (Mb)[a]	Estimated Size (Mb)	% Coverage
I	10.3	8.83	85.7
II	4.6	4.58	99.56
III	5.1	5.07	99.41
IV	5.7	5.63	98.77
V	9.2	9.13	99.23
VI	4.0	3.95	97.5
VII	4.0	3.95	97.5

[a]Chromosome sizes are derived from [48]; See [34]

Another question is how the genomic sequence compares with the integrated map, which was generated independently of the genomic sequence. The former can be generated more quickly (less than 2 years as opposed to 6 years), but with a order of magnitude increase in cost. The chromosome-specific libraries used to generate the hybridization-based physical map took 3.4 years to construct, and it took 3.4 years to carry out the clone/probe hybridization. In an alternative sequencing strategy the cosmid libraries could have been used as a sequencing resource [34]. If one compares the integrated maps of each linkage group with those of the genomic sequence at the probe level, there are 3 linkage groups in which the integrated physical map outperforms the genomic sequence in Table 4.3 in terms of its coherence with the genetic map; there are 3 linkage groups in which the genomic sequence outperforms the integrated physical map in Table 4.3; and there is one linkage group on which they perform the same (linkage group IV in Table 4.3) relative to the genetic map. Alignments of the genetic and integrated maps with the genomic sequence at the probe level for all linkage groups can be found at *http://gene.genetics.uga.edu*. With respect to long-range continuity, it appears that the competition between an integrated physical and genetic map at 23 kb resolution and the genomic sequence is a tie.

There is one other dimension on which the integrated map can be compared, the likelihood function, in Table 4.1. It can be seen that generally the integrated map outperforms the

Table 4.3: Crossing counts of lines in alignments

Linkage Group	**Genes**			**Probes**		
	G-J[a]	**J-P**[b]	**P-G**[c]	**G-J**[d]	**J-S**[e]	**S-G**[f]
1	13	0	13	18	20	19
2	11	0	11	8	7	8
3	7	0	7	7	7	10
4	8	0	8	11	12	12
5	12	0	12	13	14	13
6	8	0	8	13	10	12
7	5	0	5	13	13	12

[a]alignment genetic and integrated map at gene level
[b]alignment integrated and physical map at gene level
[c]alignment physical and genetic map at gene level
[d]alignment genetic and integrated map at probe level
[e]alignment integrated and sequence map at probe level
[f]alignment sequence and genetic map at probe level

sequence-based map, but they are very close. A similar result was found for the genetic map at the genic level [61]. In one case, the sequence-based map improves on the search for the maximum likelihood estimator of the integrated physical map.

In summary, the long-range continuity of the whole genome shotgun approach is in question beyond 9 Mb. As seen in Fig. 4.7, there can be substantial incoherence between the genomic sequence on the one hand and an integrated map on the other. Second, an integrated physical and genetic map performs comparably on this criterion in Table 4.1. Finally, there will be incoherencies between genetic and physical maps on the one hand with genomic sequence on the other hand even when a whole-genome shotgun approach is pursued to a depth of $>$ 20-fold.

4.6.9 DNA repeat organization in the *N. crassa* genome

The *N. crassa* genome consists of 10% DNA repeats as assayed by three independent methodologies ([37], [34], [24]), and 4% of these repeats are the tandemly arrayed repeats of the rDNA cluster [38]. In addition to obtaining more accurate reconstruction of the genome and a speedup in mapping with the use of chromosome-specific libraries in the physical mapping strategy in Fig. 4.2, there is the final benefit of generating a direct portrait of repeats in the genome in Fig. 4.9 as a byproduct of the physical mapping strategy in Fig. 4.2.

Each cosmid making up the integrated map has its content assayed for repeats by hybridization to all seven chromosomes of each linkage group [34]. As a consequence, each clone is classified as an S-clone, R-clone, and A-clone with the R- and A- clones usually containing DNA sequence repeats. Each clone in the integrated map can then be labeled as S, R, or A and color-coded as in Fig. 4.9 to determine the distribution of repeated sequences along the chromosome of a linkage group [52]. The third and final biological question is whether or not the repeat organization of *N. crassa* differs from that of *A. nidulans* (and that of other more complex eukaryotes) because of processes that remove repeats in the *N. crassa* genome [58].

In the *A. nidulans* genome there is an alternating pattern of DNA sequence repeats much like that in more complex eukaryotes [52]. This alternating nonrandom pattern can be quantified by the transitions between S-, R-, and A-clones along a chromosome governed by a homogeneous (across linkage groups with the exception of V) *Markov Chain* with the following transition matrix in Table 4.4:

Table 4.4: Transition Matrix

Current Clone	**Next Clone**		
	S	R	A
S	0.68	0.29	0.03
R	0.90	0.08	0.02
A	0.93	0.07	0.00

The same test of dependency were carried out for the *N. crassa* genome, and the test of first-order dependency required for description by a first order Markov Chain with its own distinct transition probabilities was significant ($X_4^2 = 84.98, P < 10^{-7}$, see Materials and Methods in [52]). The conclusion is that the repeats were non randomly distributed in the *N. crassa* genome in addition to the two known cases discussed below.

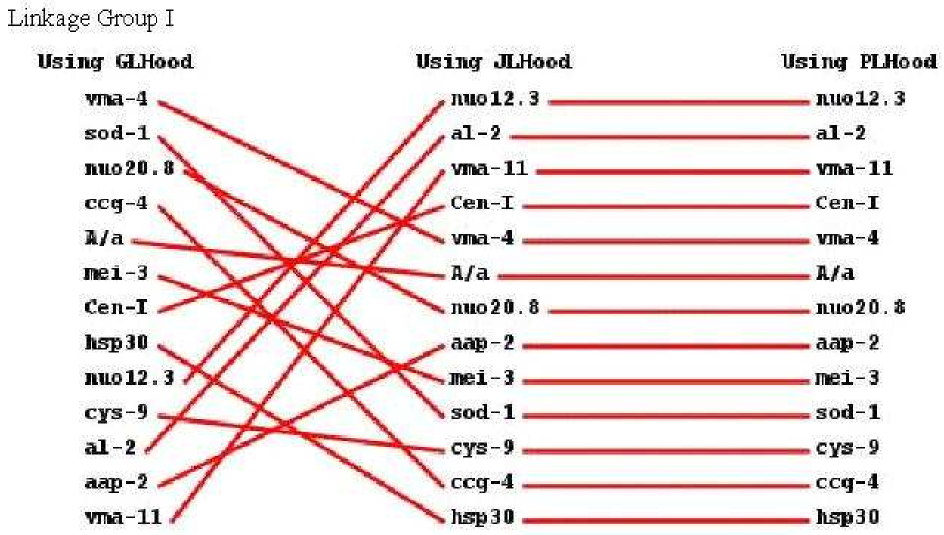
It was possible to build the integrated physical map with 70% of its clones uniquely assigned to one chromosome (being S-clones) in the integrated map. This translates into 30%

of its genome having a repetitive DNA content, which compares favorably with measurements of 10% [38], 10% [34] and 10% from the genomic sequence [24] if no more than 1/3 of each R or A clone is repetitive.

There are two known exceptions to the random distribution of DNA repeats. The *N. crassa* genome possesses 165 tandem repeats of the rDNA cluster on linkage group V [38], which can be found on the integrated map. Second, both the telomeres (which are not part of the physical map, [34]) and centromeres have repetitive DNA. Each centromere is probably on the order of 450 kb [12] and largely composed of a divergent family of repeats. If we examine the location on the integrated map, we find that they are color-coded accordingly in Fig. 4.9.

4.7 Acknowledgements

This work was supported by NSF MCB-9630910, USDA NRI Competitive Grants Program (Award No: GEO-2002-03590), and the College of Agricultural and Environmental Sciences, University of Georgia. This work is dedicated to Bob Metzenberg for the encouragement and support of this work. S. Tewari carried out algorithm development and statistical analysis as well as writing of the manuscript. J. Griffith carried out robotic arraying of libraries and clone/probe hybridizations. S. Bhandarkar executed the planning of physical mapping algorithms and writing of the manuscript. J. Arnold carried out the planning of physical mapping experiments, their statistical analysis, and writing of the manuscript.



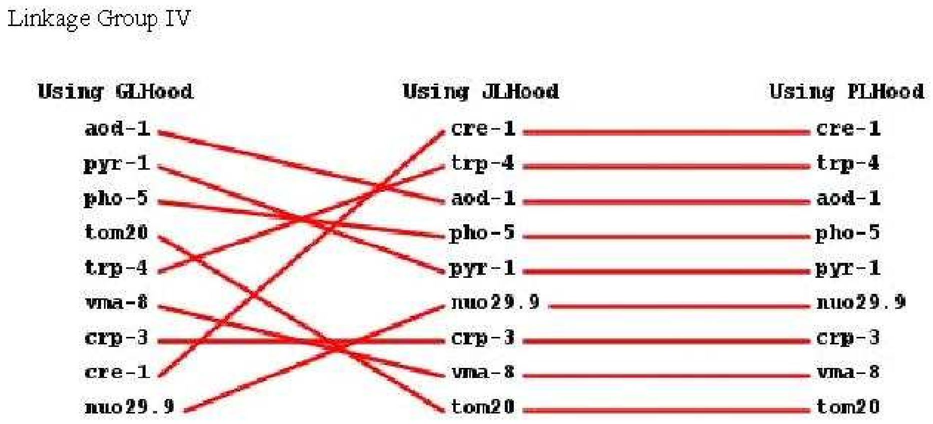
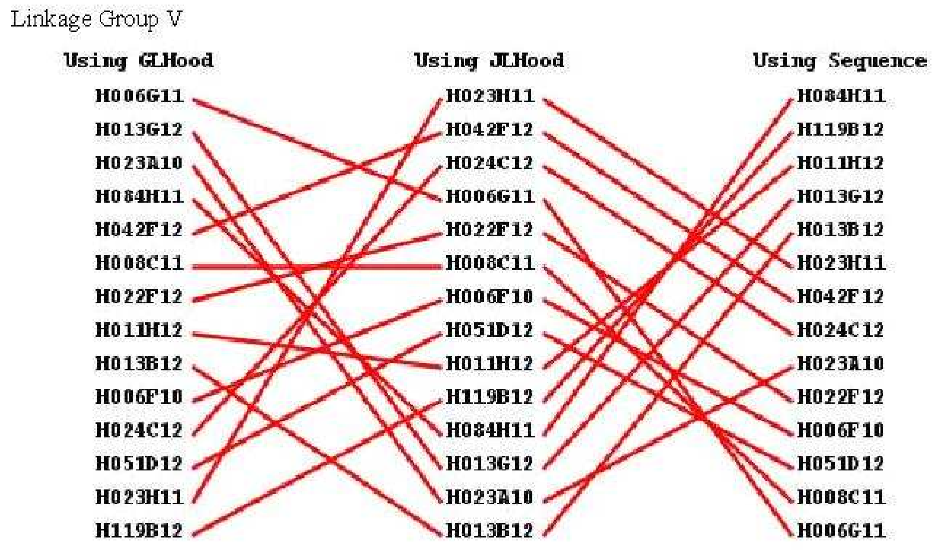


Figure 4.7: Alignment of genetic, integrated, and physical maps of linkage groups I and IV through shared markers. The orderings were produced by maximizing l_g, l_j, and l_p, respectively.



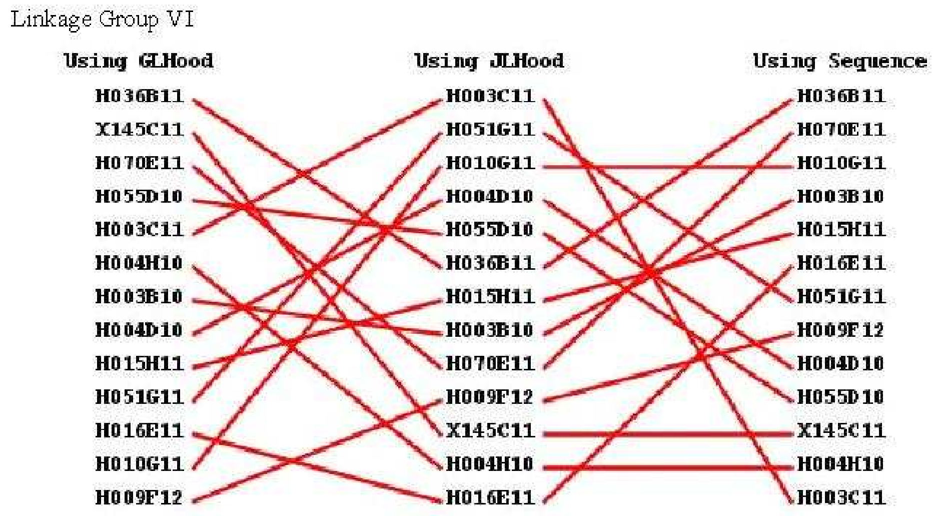


Figure 4.8: Alignment of genetic map, integrated map, and genomic sequence by probes for linkage groups V and VI. The orderings were produced by maximizing l_g and l_j and induced by genomic sequence.

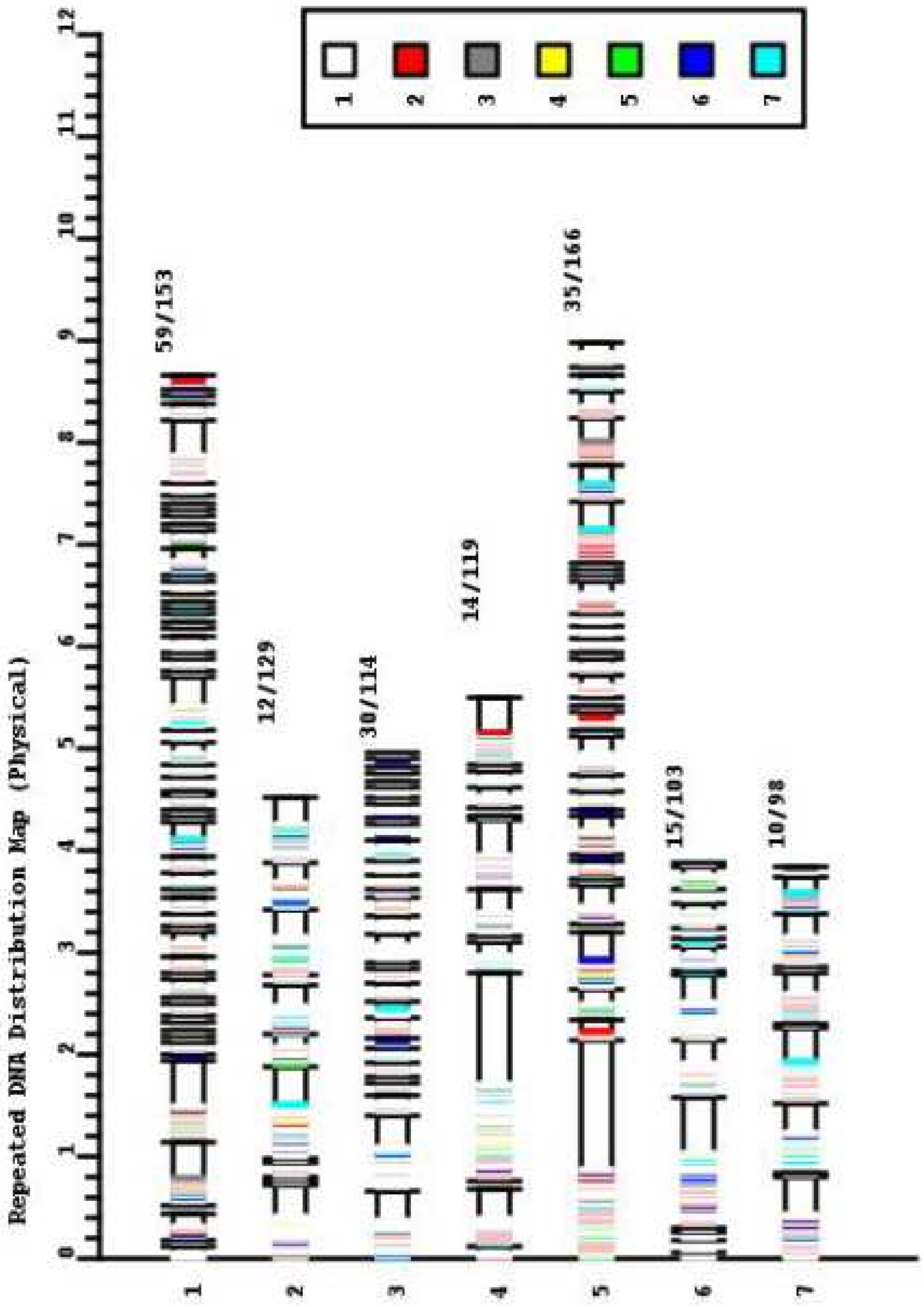


Figure 4.9: Distribution of repeated DNA sequences in *Neurospora crassa*. Physical maps of individual chromosomes are listed across the rows and contigs represented by rectangles that correspond to their estimated sizes (in megabases). Chromosomal regions containing repeated DNA sequences that are shared by 7 or fewer chromosomes are indicated by their color-coded boxes.

CHAPTER 5

CONCLUSIONS

A major methodological problem for genomics is integrating information from diverse sources [4]. Most model systems contain genetic maps, physical maps, and genomic sequences. Bioinformatics tools have been slow to develop for integrating these disparate kinds of information. In part this is explainable by the slow development of reasonable models for the experiments generating the different kinds of genomic mapping information, and even when these models are developed, the computational challenge of developing traditional inference approaches, such as the method of maximum likelihood, are quite challenging ([39]; [8]). To take the next small step of combining data sets and hence inference engines is then daunting. Here we have successfully tackled the integration of a genetic map with a hybridization-based physical map using the method of maximum likelihood.

The effort of carrying out this data integration was justified because there is a real issue of assuring the long-range continuity of a physical map, and the two distinct data sources (progeny genotypes from crosses and clone/probe hybridization data) are complementary and mutually supporting. On a fine scale of less than a map unit the physical map is more reliable, but on a large scale above a map unit the genetic map is likely to be more reliable. The two resources also represent two very different kinds of information. The genetic map carries phenotypic information, while the physical map carries the DNA sequence. Geneticists want both kinds of information on a particular trait, and an integrated map provides a means to deliver both in one package.

Here we have shown how the method of maximum likelihood provides answers to standard questions as a byproduct of its approach. With regard to the question of scale the integrated map has a scale in kilobases that is generated by the method of maximum likelihood. The scale can be attached to the map using the average insert size of each clone and the spacing between tiling probes (clones), which is part of the statistical estimation problem of specifying the physical map. Having this scale for the map allows related questions to be addressed. As shown in Fig. 4.9, what is known and the extent of the unknown (gaps in the map) are quantified. The methodology used here provides a quantitative assessment of the coherency of the information sources in Fig.s 4.7 and 4.8 as well as Table 4.3. By examining the average distance between markers on the integrated map we obtain a direct estimate of the map's resolution. Here we estimated the resolution of the integrated physical and genetic map to be 23 kb.

By combining the two maps we were able to obtain directly an elusive quantity empirically, the mapping function. We were able to relate empirically physical distance in kb to recombination distance by creating an integrated map in Fig. 4.7. There were two limitations to this mapping function. We found there was substantial variation in the physical size of a map unit along a *N. crassa* chromosome (Fig. 5.1), and there were centromere effects on the physical size of a map unit on all chromosomes of *N. crassa* when combined with the earlier finding of [12] for linkage group VII.

With such an integrated resource in hand, the integrated physical and genetic map can be used to evaluate other data sources, such as the *N. crassa* genomic sequence. Having constructed a high resolution integrated physical and genetic map, we were now in the position to evaluate the strategy of a whole genome shotgun approach. The cost of the 23 kb resolution integrated map was $468,000 over a period of 5.8 years. The cost of the $>$ 20-fold whole genome shotgun of *N. crassa* was $6,000,000 over two years. Each was generated independently.

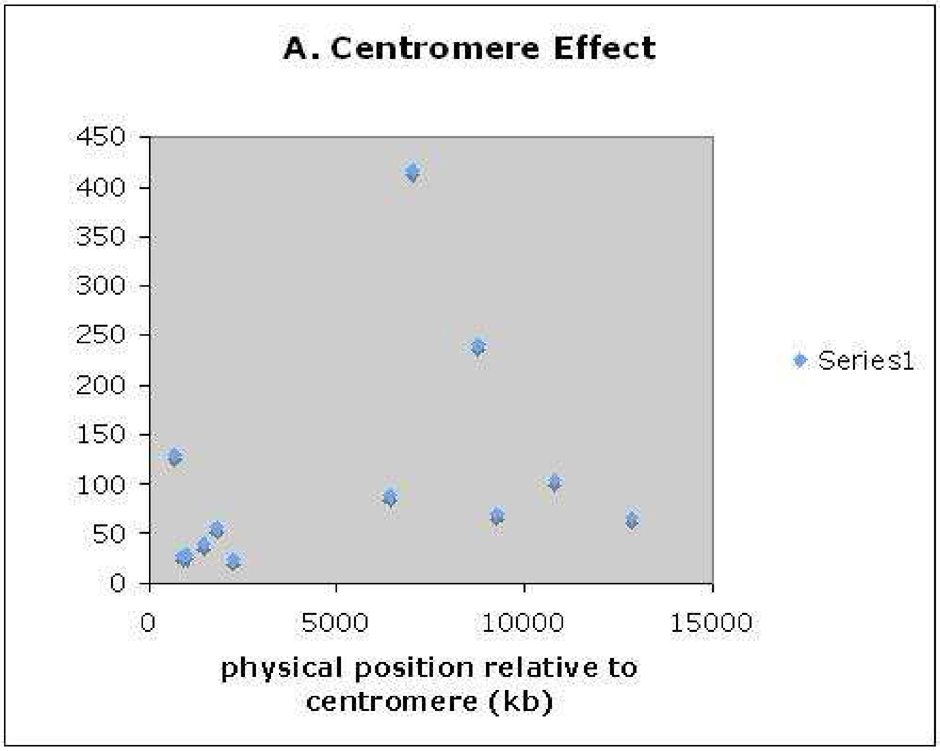
The limitations of the whole genome shotgun approach was substantial incoherence between the integrated map and genomic sequence above 9 Mb in a genome with only 10% repeats ([37]; [34]; [24]). The coherency of the integrated map vs. that of genomic sequence with the genetic map was about the same (Table 4.2). The likelihood function as a criterion did not strongly differentiate between the order of markers generated by the genomic sequence (Table 4.1) and that of the integrated physical and genetic map at 23 kb resolution.

The major purpose of generating the physical map was to address one question. How are repeats distributed in the genome? Previous work [52] had shown that the relative *A. nidulans* had a non-random alternating repeat structure much like more complex eukaryotes [52]. The question naturally arose whether or not *N. crassa* had a similar genome organization. The reason that there might be a difference from *A. nidulans* is that *N. crassa* has a process called Repeat-Induced Point mutation (or RIPing) to remove repeated DNA sequences (Selker et al., 2003). In fact most of the repeats in the genomic sequence show evidence of RIPing [24], and most of the RIPed sequences show evidence of methylation as well, which has the effect of silencing their expression. The *N.crassa* genome, in contrast to the *A. nidulans* genome, thus has a belt-and-braces strategy to control repeated DNA, such as might be introduced by a virus or other foreign DNA.

The observation is that *N.crassa* has a similar genome organization to *A.nidulans*. There is evidence for the nonrandom alternating pattern of repeated DNA sequences detected in the *A.nidulans* genome [52] as evidenced in Fig. 4.9. These are repeats in addition of the rDNA cluster on linkage group V and the centromeric/telomeric sequences. The dual strategy of RIPing and methylation appears not to be completely effective.

In summary, we have successfully addressed three fundamental questions. We have successfully integrated physical and genetic maps by the method maximum likelihood to achieve long range continuity of the physical map and to specify empirically the mapping function. Second, we have determined the limitations of the whole genome shotgun sequence approach and found substantial incoherence between an integrated map and genomic sequence above 9 Mb. Finally, we have exploited the physical mapping strategy here based on chromosome-specific libraries to show that the genome organization of DNA sequence repeats in *N.crassa* is largely non-random away from telomeres, centromeres, and the rDNA cluster.

The work here suggests two new methodological problems to be addressed in the future. The first problem has to do with genetic and physical map integration. With semi-parametric characterization of the mapping function in Fig. 5.1, it is possible to envision a constraint between the spacings and average insert size and the exchange probabilities so that the exchange probabilities and spacings are no longer independently varying parameters under the model. The semi-parametric character of the proposed mapping function could be designed to maintain one of the standard functional forms, but be allowed to vary about the standard form to account for local fluctuations in recombination rate and centromere effects. The maximization of the likelihood could then be pursued subject to this mapping function constraint using the semei-parametric mapping function. In the second problem it is clear that we can expect incoherencies between an integrated map and genomic sequence. A new method is necessary for modeling the assembly process so that a joint likelihood for crossing data, hybridization data, and sequence reads (possibly traces) simultaneously can be developed to resolve rationally these discrepancies.



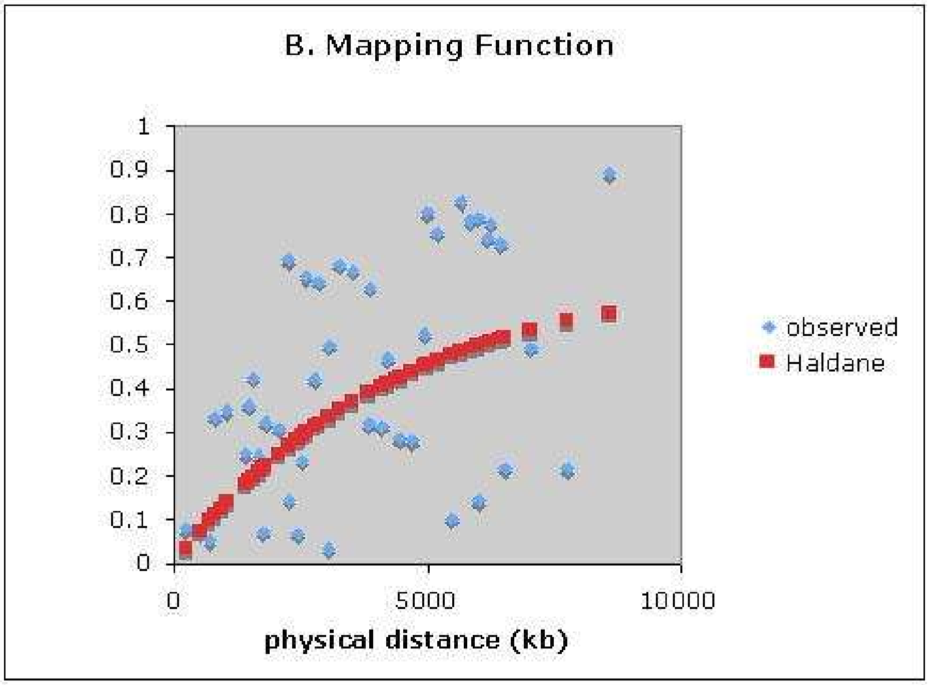


Figure 5.1: A. Plot of size of map unit in kb versus distance from centromere. B. Plot of recombination fraction r as a function physical distance d (kb) or mapping function in blue. The curve in red is a modification of the [28] mapping function to $r = (2/3)(1 - e^{(-2d/a)})$, where $a =$ (average physical distance between markers on integrated map/ average physical distance between markers on the integrated map

Bibliography

[1] A. Agresti. *Categorical Data Analysis.* Wiley-InterScience, 1990.

[2] V. Aign, U. Schulte, and J. D. Hoheisel. Hybridization-based mapping of Neurospora linkage groups ii and v. *Genetics*, 157:1015–1020, 2001.

[3] R. W. Barratt, D. Newmeyer, D. D. Perkins, and L. Garnjobst. Map construction in *Neurospora crassa. Advances in Genetics*, 6:1–93, 1954.

[4] J. Bennett and J. Arnold. *Genomics of Fungi. The Mycota VIII. Biology of the Fungal Cell*, pages 267–297. Springer-Verlag, NY, NY, 2001.

[5] S. M. Bhandarkar, J. Huang, and J. Arnold. Parallel Monte Carlo methods for physical mapping of chromosomes. Proc. IEEE Bioinformatics Conference, pages 64–75, Stanford University, Palo Alto, CA,, August 14-16 2002.

[6] S. M. Bhandarkar, J. Huang, and J. Arnold. A parallel genetic algorithms for physical mapping of chromosomes. Proc. IEEE Bioinformatics Conference, pages 567–572, Stanford University, Palo Alto, CA,, August 12-14 2003.

[7] S.M. Bhandarkar, J. Huang, and J. Arnold. An information theoretic approach to genome reconstruction, chapter:11. *Handbook of Computational Molecular Biology (S.Aluru(ed.)). CRC Press, Boca Raton, FL*, pages 11–1–11–26, 2006.

[8] S.M. Bhandarkar, S.A. Machaka, S. Shete, and R. N. Kota. Parallel computation of a maximum-likelihood estimator of a physical map. *Genetics*, 157:1021–1043, 2001.

[9] D. E .A. Catcheside. Genes in Neurospora that suppress recombination when they are heterozygous. *Genetics*, 98:55–76, 1981.

[10] D. E. A. Catcheside. A restriction and modification model for the interaction and control of recombination in *Neurospora*. *Genetic Research*, 47:157–165, 1986.

[11] E.A. Catchpole and B.J.T Morgan. Detecting parameter redundancy. *Biometrika*, 84:187–196, 1997.

[12] M. Centola and J. Carbon. Cloning and characterization of centromeric dna from *Neurospora crassa*. *MCB*, 14:1510–1519, 1994.

[13] J.G. Chakravorty and P.R. Ghosh. *Higher algebra including Modern algebra*. U.N.Dhur and Sons Private Ltd, 1996.

[14] L. Clarke and J Carbon. The structure and function of yeast centromeres. *Ann. Rev. Gen*, 19:29–55, 1985.

[15] The International HapMap Consortium. A haplotype map of the human genome. *Nature*, 437:1299–1320, 2005.

[16] R.W. Cottingham, Jr., R.M. Idury, and A.A. Sch äffer. Faster sequential genetic linkage computations. *American Journal of Human Genetics*, 53:252–263, 1993.

[17] A.J. Cuticchia, J. Arnold, and W.E. Timberlake. The use of simulated annealing in chromosome reconstruction experiments based on binary scoring. *Genetics*, 132:591–601, 1992.

[18] P. Daphne, Y. R. Seung, and R.W. Davis. Tetrad analysis possible in arabidopsis with mutation of the QUARTET(QRT) genes. *Science*, 264:1458–1460, 1994.

[19] R.H. Davis. *NEUROSPORA Contributions of a Model Organism*. Oxford University Press, 2000.

[20] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society ,Series B*, 39(1):1–38, 1977.

[21] R. W. Doerge, Z. B. Zeng, and B.S. Weir. Statistical issues in the search for genes affecting quantitative traits in experimental populations. *Statistical Science*, 12:195–219, 1997.

[22] E. Foss, R. Lande, F. W. Stahl, and C. M. Steinberg. Chiasma interference as a function of genetic distance. *Genetics*, 133:681–691, 1993.

[23] K.E. Francis, S.Y. Lam, B.D. Harrison, A.L. Bey, L.E. Berchowitz, and G.P. Copenhaver. Pollen tetrad based visual assay for meiotic recombination in ARABIDOPSIS. *PNAS,USA*, 104:3913–3918, 2007.

[24] J.E. Galagan, S.E. Calvo, K.A. Borkovich, E.U. Selker, N.D. Read, D. Jaffe, W. FitzHugh, L. Ma, S. Smirnov, S. Purcell, B. Rehman, T. Elkins, R. Engels, S. Wang, C.B. Nielsen, J. Butler, M. Endrizzi, D. Qui, P. Ianakiev, D. Bell-Pedersen, M.A. Nelson, M. Werner-Washburne, C.P. Selitrennikoff, J.A. Kinsey, E.L. Braun, A. Zelter, U. Schulte, G.O. Kothe, G. Jedd, W. Mewes, C. Staben, E. Marcotte, D. Greenberg, A. Roy, K. Foley, J. Naylor, N. Stange-Thomann, R. Barrett, S. Gnerre, M. Kamal, M. Kamvyssells, E. Mauceli, C. Bielke, S. Rudd, D. Frishman, S. Krystofova, C. Rasmussen, R.L. Metzenberg, D.D. Perkins, S. Kroken, C. Cogoni, G. Macino, D. Catcheside, W. Li, R.J. Pratt, S.A. Osmani, C.P.C. DeSouza, L. Glass, M.J. Orbach, J.A. Berglund, R. Voelker, O. Yarden, M. Plamann, S. Seller, J.C. Dunlap A. Radford, R. Aramayo, D.O. Natvig, L.A. Alex, G. Mannhaupt, D.J. Ebbole, M. Freitag, I. Paulsen, M.S. Sachs, E.S. Lander, C. Nusbaum, and B. Birren. The genome sequence of the filamentous fungus *Neurospora crassa*. *Nature*, 422:859–868, 2003.

[25] N.H. Giles, De Serres J. Frederick, Jr., and Barbour E. Studies with purple adenine mutants in Neurospora crassa. ii. tetrad analyses from a cross of an ad-3a mutant with an ad-3b mutant. *Genetics*, pages 608–617, 1957.

[26] A. et al. Goffeau. Life with 6000 genes. *Science*, 274:546–567, 1996.

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

[28] J.B.S. Haldane. The combination of linkage values and the calculation of distance between the loci of linkage factors. *Journal of Genetics*, 8:229–309, 1919.

[29] D. Hall, S.M. Bhandarkar, and J. Wang. ODS2: A multiplatform software application for creating integrated physical and genetic maps. *Genetics*, 157:1045–1056, 2001.

[30] J Huang and S. M . Bhandarkar. A comparison of physical mapping algorithms based on the maximum likelihood model. *Bioinformatics*, 19(11):1303–1310, 2003.

[31] R.M. Idury and R.C. Elston. A faster and more general hidden markov model algorithm for multipoint likelihood calculations. *Human Heredity*, 47:197–202, 1997.

[32] L. John and L. William. *Java Software Solutions*. Oxford University Press, 2004.

[33] J. Kececioglu, S. Shete, and J. Arnold. Reconstructing order and distance in physical maps using non-overlapping probes. *RECOMB*, pages 183–192, 2000.

[34] H. S. Kelkar, J. Griffith, M. E. Case, S. F. Covert, R. D. Hall, C. H. Keith, J. S. Oliver, M. J. Orbach, M. S. Sachs, J. R. Wagner, M. J. Weise, J. K. Wunderlich, and J. Arnold. The *Neurospora crassa* genome: Cosmid libraries sorted by chromosome. *Genetics*, 157:979–990, 2001.

[35] S. Kirkpatrick, C.D. Jr. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[36] K. J. Kochut, J. Arnold, A. Sheth, J. Miller, E. Kraemer, B. Arpinar, and Cardoso. J. A distributed workflow system for discovering protein-protein interactions. *Parallel and Distributed Databases*, 13:43–72, 2003.

[37] R. Krumlauf and G. A. Marzluf. Characterization of the sequence complexity and organization of the*Neurospora crassa* genome. *Biochemistry*, 18(17):37053713, 1979.

[38] R. Krumlauf and G. A. Marzluf. Genomic organization and characterization of the repetitive and inverted repeat dna sequences in*Neurospora crassa*. *Jornal of Biological Chemistry*, 255:1138–1145, 1980.

[39] E.S. Lander and P. Green. Construction of multi-locus genetic linkage maps in humans. *Proc.Natl.Acad.Sci.USA*, 84:2363–2367, 1987.

[40] E.S Lander and N.J Schork. Genetic dissection of complex traits. *Science*, 265:2037–2048, 1994.

[41] K. Lange, H. Zhao, and T. P. Speed. The Poisson-skip model of crossing-over. *The Annals of Applied Probability*, 7:299–313, 1997.

[42] K. Mather. *The Measurement of Linkage in Heredity.* John Wiley & Sons NY NY, 1951.

[43] R. B. Meagher, M. D. McLean, and J. Arnold. Recombination within a subclass of restriction fragment polymorphisms may help link classical and molecular genetics. *Genetics*, 120:809–818, 1988.

[44] X-L Meng and D.B. Rubin. Using EM to obtain asymptotc variance-covariance matrices: The SEM algorihm. *Journal of the American Statistical Association*, 86:899–909, 1991.

[45] D. Mester, Y. Romin, D. Minkov, E. Nevo, and A. Korol. Constructing large-scale genetic maps using an evolutionary strategy algorithm. *Genetics*, 165:2269–2282, 2003.

[46] M.A. Nelson, M.E. Crawford, and D.O. Natvig. Restriction polymorphism maps of *Neurospora crassa*: 1998 update. *http://www.fgsc.net/fgn45/45rflp.html*, 1998.

[47] L. J. Norman, S. Kotz, and W. K. Adrienne. *Univariate Discrete Distributions.* Wiley-Interscience, 2 edition, 1993.

[48] M. J Orbach. One liners. *Fungal Genetics*, 39:92, 1992.

[49] M. J. Orbach. A cosmid with a H_yR marker for fungal library construction and screening. *Gene*, 150:159–162, 1994.

[50] D. D. Perkins, A. Radford, D. Newmeyer, and M Bjorkman. Chromosomal loci of *Neurospora crassa. Microbiology*, 46:426–570, 1982.

[51] D. D. Perkins, A. Radford, and M. S . Sachs. *The Neurospora Compendium, Chromosomal Loci.* Academic Press, 2001.

[52] R. A. Prade, J. Griffith, K. Kochut, J. Arnold, and W. E . Timberlake. *In vitro* reconstruction of the *Aspergillus (=Emericella) nidulans*. *Proc. Natl. Acad. Sci. USA*, 94:14564–14569, 1997.

[53] N.B. Raju. Meiosis and ascospore genesis in Neurospora. *Eur.J.Cell.Biol.*, 23:208–223, 1980.

[54] C.R. Rao. *Linear Statistical Inference and Its Application.* Wiley-Interscience; 2 edition, 2002.

[55] M.G . Schechtman. Isolation of telomere DNA from *Neurospora crassa. Mol. Cell. Biol*, 7:3168–3177, 1987.

[56] S.J. Schwager, M.A. Mutschler, W.T. Federer, and B.T. Scully. The effect of linkage on sample size determination for multiple trait selection. *Theoretical and Applied Genetics / Theoretische und angewandte Genetik* ., 86:964–974, 1993.

[57] S.R. Searle. *Matrix Algebra Useful for Statistics.* John Wiley & Sons, 1982.

[58] E. U. Selker, N. A. Tountas, S. H. Cross, B. S. Margolin, J. G. Murphy, A. P. Bird, and M. Freitag. The methylated component of the *Neurospora crassa* genome. *Nature*, 422:893–897, 2003.

[59] S.D. Silvey. *Statistical Inference.* Chapman and Hall:London, 1975.

[60] A.H. Sturtevant. The linear arrangement of six sex-linked factors in *Drosophila* as shown by their mode of association. *J.Exp.Zool*, 14:43–49, 1913.

[61] S. Tewari, J. Arnold, and S.M. Bhandarkar. Likelihood of a particular order of genetic markers and construction of genetic maps. *Journal of Bioinformatics and Computational Biology*, 6(1):125–162, 2008.

[62] S. Tewari, S.M. Bhandarkar, and J. Arnold. Efficient recursive linking algorithm for computing the likelihood of an order of a large number of genetic markers. In P. Markstein and Y. Xu, editors, *Computational Systems and Bioinformatics Conference*, volume 4 of *Advances in Bioinformatics and Computational Biology*, pages 191–198, Stanford University, Palo Alto, CA, August 2006. Life Sciences Society, Imperial College Press.

[63] S. Tewari, S.M. Bhandarkar, and J. Arnold. Design and analysis of an efficient recursive linking algorithm for constructing likelihood based genetic maps for a large number of markers. *Journal of Bioinformatics and Computational Biology*, 5(2(a)):201–250, 2007.

[64] H.C. Thomas, E.L. Charles, L.R. Ronald, and S. Clifford. *Introduction to Algorithms.* The MIT Press; 2 edition, 2001.

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

[66] Y. Wang, R. A. Prade, J. Griffith, W. E. Timberlake, and J. Arnold. A fast random cost algorithm for physical mapping. *Proc. Natl. Acad. Sci. USA*, 91:11094–11098, 1994.

[67] D.E. Weeks and K. Lange. Preliminary ranking procedures for multilocus ordering. *Genomics*, 1:236–242, 1987.

[68] C.J. Williams, W.W. Anderson, C.J. Brown, and J. Arnold. An analysis of density-dependent viability selection. *Journal of the American Statistical Association*, 84:662–668, 1989.

[69] B.J. Winer. *Statistical Principles in Experimental Desgin.* McGraw-Hill,New York, 2 edition, 1971.

[70] E.A. Winzeler, D.R. Richards, A.R. Conway, A.L. Goldstein, S. Kalman, M.J. McCullough, J.H. McCusker, D.A. Stevens, L. Wodicka, D.J. Lockhart, and R.W. Davis. Direct allelic variation scanning of the yeast genome. *Science*, 281:1194–1197, 1998.

[71] M. Q. Zhang and T. G . Marr. Genome mapping by nonrandom anchoring: a discrete theoretical analysis. *Proc. Natl. Acad. Sci. USA*, 90:600–604, 1993.

[72] H. Zhao, M.S. McPeek, and T. P. Speed. Statistical analysis of chromatid interference. *Genetics*, 139:1057–1065, 1995.

[73] H. Zhao, M.S. McPeek, and T. P. Speed. Statistical analysis of crossover interference using the Chi-square model. *Genetics*, 139:1045–1056, 1995.

[74] H. Zhao and T. P. Speed. Stochastic modeling of the crossover process during meiosis. *Communications in Statistics, Theory and Methods*, 27:1557–1580, 1998.

[75] H. Zhao and T.P. Speed. Statistical analysis of ordered tetrads. *Genetics*, 150:459–472, 1998.