

UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Computational methods for analyzing human genetic variation

Permalink

<https://escholarship.org/uc/item/4911n54v>

Author

Bansal, Vikas

Publication Date

2008

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA, SAN DIEGO

Computational Methods for Analyzing Human Genetic Variation

A dissertation submitted in partial satisfaction of the requirements for the degree
Doctor of Philosophy

in

Computer Science and Engineering

by

Vikas Bansal

Committee in charge:

Professor Vineet Bafna, Chair
Professor Ramamohan Paturi
Professor Pavel Pevzner
Professor Nicholas Schork
Professor Glenn Tesler

2008

Copyright
Vikas Bansal, 2008
All rights reserved.

The dissertation of Vikas Bansal is approved and it is acceptable in quality and form for publication on microfilm:

Chair

University of California, San Diego

2008

DEDICATION

to my parents

TABLE OF CONTENTS

Signature Page	iii
Dedication	iv
Table of Contents	v
List of Figures	viii
List of Tables	x
Acknowledgements	xi
Vita	xiii
Abstract of the Dissertation	xv
Chapter 1 Introduction	1
1.1 Single Nucleotide Polymorphisms (SNPs)	4
1.1.1 SNPs and Haplotypes	5
1.2 Meiotic Recombination	6
1.2.1 Recombination Rates	7
1.2.2 Recombination and Linkage Disequilibrium (LD)	8
1.3 Reconstructing Evolutionary Histories from Haplotype data	10
1.3.1 Counting Recombination Events	13
1.4 Finding Inversions using SNP haplotype data	14
1.5 The Haplotype Assembly Problem	15
Chapter 2 Counting Recombination Events: Lower bounds and Recombination Hotspots	18
2.1 Preliminaries	22
2.1.1 Lower Bounds on the Minimum Number of Recombination Events	22
2.1.2 Combining Local Recombination Bounds	23
2.2 Bounds based on Haplotype Diversity	24
2.2.1 The lower bound R_g	25
2.2.2 Comparison of the lower bound R_g with previous bounds	26
2.2.3 Bounds for Haplotypes with Missing Data	30
2.2.4 Application to Haplotype Data from LPL locus	31
2.2.5 Application of Lower Bounds to reveal Recombination Hotspots	32
2.3 History Based Lower Bounds	35
2.3.1 Recombinant Intermediates and the bound R_I	36
2.3.2 Computing Recombinant Intermediates	40
2.3.3 Results for R_I bound	41

2.4	Discussion and Future Work	41
2.5	Acknowledgement	44
Chapter 3	Counting Recombination Events: Conflict Graph and Lower Bounds . .	45
3.1	Definitions	46
3.2	Connected Components in the Conflict Graph	47
3.2.1	Extensions to the R_c lower bound	57
3.3	Comparison of R_c with other bounds	60
3.3.1	Application to a Drosophila Dataset	66
3.4	Acknowledgement	67
Chapter 4	Detecting large inversions from whole-genome SNP haplotype data . .	68
4.1	Introduction	68
4.2	Methods	73
4.2.1	Using LD in population data to detect inversions	74
4.2.2	The Inversion Statistic	75
4.2.3	Measuring LD	76
4.2.4	Defining multi-SNP markers	77
4.2.5	Simulating Inversions	78
4.3	Results	79
4.3.1	Power to detect Inversion Polymorphisms	79
4.3.2	Scanning the HapMap data for inversion polymorphisms	80
4.3.3	Sequence Analysis of Inversion Breakpoints	85
4.3.4	Assessing the false positive rate	88
4.4	Discussion	91
4.5	Appendix	93
4.5.1	Haplotype Data	93
4.5.2	Identifying potential inversions	94
4.5.3	Sequence Analysis	95
4.5.4	Coalescent Simulations	96
4.6	Acknowledgements	97
Chapter 5	Haplotype Assembly from Whole-genome sequence data	98
5.1	Introduction	98
5.2	Methods	103
5.2.1	Haplotype Likelihood	104
5.2.2	Markov chain Monte Carlo algorithm	106
5.2.3	Choosing Γ	108
5.2.4	A graph-partitioning approach	110
5.2.5	The complete MCMC algorithm	112
5.3	MEC score for haplotype assembly and posterior error probabilities . .	115
5.4	Results	116
5.4.1	HuRef sequence data	116

5.4.2	Performance of HASH on Simulated data	118
5.4.3	HASH versus other MCMC algorithms	121
5.4.4	Haplotypes for HuRef	122
5.4.5	Estimating accuracy of HuRef haplotypes	124
5.5	Discussion	128
5.6	Acknowledgements	131
Chapter 6	A Combinatorial Algorithm for the Haplotype Assembly Problem . . .	132
6.1	Introduction	132
6.2	Methods	133
6.2.1	Preliminaries and Optimization	133
6.2.2	MEC optimization using max-cuts	135
6.2.3	Assigning weights to edges of $G_X(H)$	137
6.2.4	Computing Max-Cuts	138
6.3	Results	139
6.3.1	MEC scores for HuRef chromosomes	140
6.3.2	Simulations using HuRef data	142
6.4	Switch error rate of HuRef haplotypes	144
6.5	Acknowledgements	147
Chapter 7	Markov chains for Haplotype Assembly: Mixing time analysis	148
7.1	Introduction	148
7.2	Mixing time of Markov chains	151
7.2.1	Conductance and Mixing time	151
7.2.2	The Coupling Argument	152
7.2.3	The canonical paths argument	153
7.2.4	Decomposition Theorem for analyzing mixing time	154
7.3	A Family of Fragment Matrices	155
7.3.1	$\mathcal{M}(\Gamma_1)$ has poor mixing time	158
7.4	A Markov chain with polynomial mixing time	158
7.4.1	Mixing time of the Markov chain restricted to D^k	159
7.4.2	Mixing time of the projection Markov chain	167
References	170

LIST OF FIGURES

Figure 1.1: Illustration of the two copies from a region of a chromosome present in three individuals.	6
Figure 1.2: Illustration of a recombination event between the two chromosomes in an individual.	7
Figure 1.3: Example of an Ancestral Recombination Graph explaining a set S of binary sequences with two recombination nodes.	12
Figure 2.1: Distribution of the R_g lower bound and the number of distinct haplotypes for different recombination rates.	29
Figure 2.2: The density of detected recombination events in a 216KB segment of the class II region of MHC and a 206KB region on human chromosome 1.	33
Figure 2.3: Illustration of a example for which R_s is sub-optimal.	37
Figure 3.1: Illustration of how the conditions of Lemma 3 constrain the matrix to have particular values.	50
Figure 3.2: The structure of the matrix M for which $R_h = 2/3 \cdot R_c$	62
Figure 3.3: Haplotype matrix for ADH locus of <i>Drosophila</i>	66
Figure 4.1: Haplotype patterns for the 900kb inversion on chromosome 17 in the CEU HapMap sample	71
Figure 4.2: Unusual Linkage Disequilibrium observed in SNP data when the inverted haplotype (w.r.t the reference sequence) has very high frequency	74
Figure 4.3: Power of the inversion statistic to detect inversions of varying length and frequency	80
Figure 4.4: Genomic overview of a 1.4 Mb region at 16p12 predicted to have an inversion in both the CEU and YRI ‘analysis panels’	82
Figure 4.5: Overview of a ≈ 1.2 Mb long predicted inversion on chromosome 10	83
Figure 4.6: Overlap of predicted inversion on chromosome 6 with TCBA1 gene.	87
Figure 4.7: Splice isoforms of the ICAp69 gene are approximately consistent with a predicted YRI inversion breakpoint on chromosome 7.	88
Figure 4.8: Length distribution of predicted inversions in the YRI ‘analysis panel’	90
Figure 5.1: Illustration of how haplotypes can be assembled from sequenced reads.	100
Figure 5.2: Example of a fragment matrix for which two haplotypes have equal likelihood.	107
Figure 5.3: Illustration of the recursive graph-partitioning algorithm for computing Γ	113
Figure 5.4: An example of a fragment matrix for a haplotype segment from chromosome 22 of HuRef.	114
Figure 5.5: Distribution of the number of variants among haplotypes of different sizes.	118

Figure 5.6: Comparison of the switch error rate for HASH and the MCMC algorithm with Γ_1	119
Figure 5.7: Fraction of variant calls with a high posterior error probability using the HASH algorithm.	120
Figure 5.8: Results of running the MCMC algorithm with different Γ on a fragment matrix with $n = 200$ columns (from chromosome 22 of HuRef genome).	122
Figure 5.9: MEC score for three different methods for HuRef chromosomes.	123
Figure 5.10: Fraction of variant calls with high posterior error probability versus sequencing error probabilities for HuRef chromosome 22.	125
Figure 5.11: Comparison of haplotypes assembled using sequence data with the preferred HapMap phasing for each pair of adjacent SNPs inferred from the HapMap haplotypes.	127
Figure 5.12: Mismatch Rate and the “Adjusted Mismatch Rate” (Error Rate) of the HuRef haplotypes estimated by comparison with the CEU HapMap haplotypes.	129
Figure 6.1: Distribution of the number of variant calls per fragment and the span of fragments.	141
Figure 6.2: Comparison of HuRef MEC scores for four methods.	142
Figure 6.3: Comparison of simulated vs estimated errors and plot of haplotype switch error as a function of depth of coverage.	143
Figure 6.4: Haplotype log-likelihood curves for four different values of the switch error rate.	145
Figure 6.5: The log-likelihood curve for the HuRef haplotypes for chromosome 22.	146
Figure 7.1: A fragment matrix $\mathcal{X}_{d,n}$ ($n = 20, d = 2$ as shown) for which two haplotypes H_1 and H_2 have equal likelihood.	156

LIST OF TABLES

Table 2.1: Properties of three lower bounds and two summary statistics for samples generated using the coalescent.	27
Table 2.2: The number of detected recombination events using bounds for missing data for the LPL datasets	32
Table 2.3: Comparison of the R_s and R_I lower bounds for datasets from the SeattleSNP project.	42
Table 4.1: List of predicted inversions with some evidence supporting the inverted orientation.	86

ACKNOWLEDGEMENTS

I have been fortunate to have Vineet as my adviser. He has given me freedom to choose research problems and at the same time worked closely with me. I cannot thank him enough for his guidance, support and encouragement through these years.

I would like to thank Professor Pavel Pevzner for his guidance and encouragement. I am greatly obliged to Aaron Halpern and Samuel Levy for their willingness to work with us. During my graduate years, I have learned a lot about Bioinformatics and research in general through interactions with fellow students and postdocs. I would like to thank all members of the Bioinformatics group (past and present), in particular, Ali, Alkes, Ari, Ben, Dima, Degui, Mark, Neil, Nitin, and Shaojie. It has been a pleasure to get to know fellow students from India during my stay at UCSD: Suchit, Saurabh, Manish, Mac, Ankit, Nitin, Gaurav, Ragesh and many others. Special thanks are due to Saurabh and Gaurav for innumerable entertaining discussions.

Lastly, I am forever indebted to my parents for their love and support and encouragement in every endeavor I have pursued.

Chapter 2, in full, was published in the *Journal of Computational Biology*, Vol 13(2), pp 501-21, 2006, V. Bafna and V. Bansal, "Inference about Recombination from Haplotype Data: Lower Bounds and Recombination Hotspots". The dissertation author was the primary investigator and author of this paper.

Chapter 3, in full, was published in *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol 1(2), pp 78-90, 2004, V. Bafna and V. Bansal, "The Number of Recombination Events in a Sample History: Conflict Graph and Lower Bounds". The dissertation author was the primary investigator and author of this paper.

Chapter 4, in full, was published in *Genome Research*, Vol 17(2), pp 219-30, 2007, V. Bansal, A. Bashir and V. Bafna, "Evidence for large inversion polymorphisms in the human genome from HapMap data". The dissertation author was the primary investigator and author of this paper.

Chapter 5, in full, will be published in Genome Research, V. Bansal, A. Halpern, N. Axelrod, V. Bafna, “An MCMC Algorithm for Haplotype Assembly from Whole-genome Sequence Data”. The dissertation author was the primary investigator and author of this paper.

Chapter 6, in full, will appear in the Proceedings of the European Conference on Bioinformatics, 2008, V. Bansal and V. Bafna, “HapCUT: an Efficient and Accurate Algorithm for the Haplotype Assembly Problem”. The dissertation author was the primary investigator and author of this paper.

VITA

- 2003 Bachelor of Technology in Computer Science and Engineering, Indian Institute of Technology, Delhi
- 2006 Master of Science in Computer Science and Engineering, University of California, San Diego
- 2008 Doctor of Philosophy in Computer Science and Engineering, University of California, San Diego

PUBLICATIONS

“An MCMC Algorithm for Haplotype Assembly from Whole-genome Sequence Data”. Vikas Bansal, Aaron L. Halpern, Nelson Axelrod and Vineet Bafna. *Genome Research*, to appear, 2008

“HapCUT: An Efficient and Accurate Algorithm for the Haplotype Assembly Problem”. Vikas Bansal and Vineet Bafna. to appear in the Proceedings of the European Conference on Computational Biology, 2008.

“The Diploid Genome Sequence of an Individual Human”. S Levy, G Sutton, PC Ng, L Feuk, AL Halpern, et al. *PloS Biology* 5(10), 2007

“Evidence for large inversion polymorphisms in the human genome from HapMap data”. Vikas Bansal, Ali Bashir and Vineet Bafna. *Genome Research* 17:219-230, 2007

“A Decomposition Theory for Phylogenetic Networks and Incompatible Characters”. Dan Gusfield, Vikas Bansal, Vineet Bafna and Yun S. Song. *Journal of Computational Biology*, Vol 14: pp 1247-1272, 2007.

“Inference about Recombination from Haplotype Data: Lower Bounds and Recombination Hotspots”. Vineet Bafna and Vikas Bansal. *Journal of Computational Biology*, Vol 13(2): pp 501-21, 2006.

“Improved Recombination Lower Bounds for Haplotype Data”. Vineet Bafna and Vikas Bansal. In Proceedings of the Ninth Annual International Conference on Research in Computational Molecular Biology (RECOMB), pp. 569-584, 2005

“ Fundamental Decomposition Theory for Phylogenetic Networks and Incompatible Characters”.

Dan Gusfield and Vikas Bansal. In Proceedings of the Ninth Annual International Conference on Research in Computational Molecular Biology (RECOMB), pp. 217-232, 2005

“The Number of Recombination Events in a Sample History: Conflict Graph and Lower Bounds”.

Vineet Bafna and Vikas Bansal. IEEE/ACM Trans. on Comp. Biology and Bioinformatics, Vol 1(2): pp 78-90, April-June 2004.

“Labeling Smart Dust”.

Vikas Bansal, Freidhelm M. Heide, and Christian Sohler. 12th Annual European Symposium of Algorithms, 2004.

FIELDS OF STUDY

Major Field: Computer Science and Engineering
Bioinformatics.
Vineet Bafna

ABSTRACT OF THE DISSERTATION

Computational Methods for Analyzing Human Genetic Variation

by

Vikas Bansal

Doctor of Philosophy in Computer Science and Engineering

University of California, San Diego, 2008

Professor Vineet Bafna, Chair

In the post-genomic era, several large-scale studies that set out to characterize genetic diversity in human populations have significantly changed our understanding of the nature and extent of human genetic variation. The International HapMap Project has genotyped over 3 million Single Nucleotide Polymorphisms (SNPs) in 270 humans from four populations. Several individual genomes have recently been sequenced and thousands of genomes will be available in the near future. In this dissertation, we describe computational methods that utilize these datasets to further enhance our knowledge of the fine-scale structure of human genetic variation. These methods employ a variety of computational techniques and are applicable to organisms other than human.

Meiotic recombination represents a fundamental mechanism for generating genetic diversity by shuffling of chromosomes. There is great interest in understanding the non-random distribution of recombination events across the human genome. We describe combinatorial methods for counting historical recombination events using population data. We demonstrate that regions with increased density of recombination

events correspond to regions identified as recombination hotspots using experimental techniques.

In recent years, large scale structural variants such as deletions, insertions, duplications and inversions of DNA segments have been revealed to be much more frequent than previously thought. High-throughput genome-scanning techniques have enabled the discovery of hundreds of such variants but are unable to detect balanced structural changes such as inversions. We describe a statistical method to detect large inversions using whole genome SNP population data. Using the HapMap data, we identify several known and putative inversion polymorphisms.

In the final part of this thesis, we tackle the haplotype assembly problem. High-throughput genotyping methods probe SNPs individually and are unable to provide information about haplotypes: the combination of alleles at SNPs on a single chromosome. We describe Markov chain Monte Carlo (MCMC) and combinatorial algorithms for reconstructing the two haplotypes for an individual using whole genome sequence data. These algorithms are based on computing cuts in graphs derived from the sequenced reads. We analyze the convergence properties of the Markov chain underlying our MCMC algorithm. We apply these methods to assemble highly accurate haplotypes for a recently sequenced human.

Chapter 1

Introduction

The sequencing of the human genome in 2001 marked the beginning of a new era in human genetics and biomedical research. With the availability of a reference human genome sequence, it has become possible to catalog differences between individual genomes. It is well known that differences in DNA are responsible for a substantial fraction of phenotypic variation in humans and that a comprehensive understanding of human genetic variation will enable the discovery of genetic variants responsible for increased susceptibility to various diseases. This genetic variation is present in many different forms and sizes: in the form of single letter changes known as Single Nucleotide Polymorphisms (SNPs) but also as insertions, deletions and inversions of DNA segments several thousand to millions of base pairs long.

During the sequencing and analysis of the human genome, more than a million candidate SNPs were discovered by comparison of the genomic sequence of a few individuals (Sachidanandam et al., 2001). Given the huge number of SNPs in the human genome and the relative ease of obtaining SNP information, the International HapMap project (The International HapMap Consortium, 2003) was formed in 2002 to catalog genetic variation at SNPs in human populations. The HapMap project focused on common SNPs, SNPs where both letters are present in a population above a minimal frequency (1% of the individuals). The main goal of this project was to determine

the patterns of common genetic variation in human populations which would enable the efficient design of genome-wide association studies for finding the genetic basis of complex diseases. The HapMap project was completed in 2007 and has generated genotypes for more than 3 million Single Nucleotide Polymorphisms (SNPs) have been genotyped in 270 humans from four different populations. The HapMap data has proved useful in understanding recombination patterns and identification of genes under positive selection in the human genome(The International HapMap Consortium, 2005). In the post-HapMap era, several commercial SNP genotyping chips have been developed, some of which can interrogate up to a million Single Nucleotide Polymorphisms (SNPs) in the human genome. The availability of these chips has enabled whole genome association studies where one can compare the DNA sequence of thousands of healthy and diseased individuals to identify genetic variants that are associated with the disease. In the past year alone, association studies for common diseases such as diabetes, coronary artery disease, etc and physiological traits such as height and eye color have identified hundreds of SNPs associated with increased risk for diseases and that can explain the variation in traits.

The HapMap project and other projects (see e.g. Perlegen study (Hinds et al., 2005)) have significantly enhanced our understanding of the patterns of variation at SNPs in human populations. In comparison, until recently, very little was known about large scale genomic variants such as deletions, insertions, duplications (copy-number changes), and inversions. Knowledge about the location of these large scale variants, collectively referred to as “structural variation”, has recently started to accumulate. High-throughput techniques based on comparative hybridization which compare the intensity of genomic segments between individuals have allowed biologists to discover thousands of structural variants, in particular copy number polymorphisms (a segment of DNA present in a variable number of copies in different individuals). Some of these have also been linked to human phenotype variation Sebat et al. (2004); Lucito et al. (2003); Iafrate et al. (2004). Structural genetic variants have the potential to affect phenotypes through multiple mechanisms such as gene deletion/disruption, gene fusions, changes

in gene copy number, increasing/decreasing distance between functional elements in the genome through inversions, etc. However, relative to SNPs, the catalog for some of these polymorphisms is far from being complete. Furthermore, intensity based techniques are limited by their inability to discover balanced structural variants such as inversions or insertions of previously unknown sequences in the genome. High throughput sequencing represents a direct way for discovery of all kinds of structural variants including copy-neutral variations such as inversions. Tuzun et al. (2005) mapped paired-end sequence data from large-insert clones (40 kb) from an individual genome to the reference human genomic sequence to reveal sites of deletions, insertions and inversions. This strategy has been applied to several different individual genomes to reveal many more such variants. In order to obtain a comprehensive catalog of all forms of genetic variants, complete sequencing of many individual genomes represents the ideal strategy. In the post-HapMap era, advancements in sequencing technology are driving down the cost of sequencing and projects that aim to sequence hundreds of individual genomes have been launched (see e.g the 1000 Genomes Project: www.1000genomes.org). In 2007, several individuals such as J. Craig Venter and James Watson have had their complete genomes sequenced.

The wealth of SNP population data generated by projects such as the HapMap and the DNA sequence data from individual genome sequencing projects contains useful information about human evolutionary history and the fine-scale structure of human genetic variation. Lot of this information can be obtained by simple computational analysis. However, sophisticated computational methods especially those based on modeling human genetic variation have the potential to uncover hidden information and make more accurate population-genetic inferences. In this dissertation, we address three computational problems related to analysis of human genetic variation data:

- Counting historical recombination events using population haplotypes
- Reconstruction of individual haplotypes using DNA fragments from genome sequencing data

- Discovery of large inversions in the human genome using whole genome SNP haplotype data

The first two problems can be formulated as self-contained computational problems. We have attempted to explore the computational complexity of these two problems and obtain efficient algorithms that work well on real datasets. Using whole genome SNP haplotype data to discover large inversions represents a novel use of such data. We address this problem in a statistical framework where we model the effect of large inversions on SNP haplotype patterns and scan the genome for potential inversion breakpoints using a simple score designed to capture deviations from expected haplotype patterns. In the rest of this chapter, we give some background on SNPs, haplotypes and the process of recombination. We also present the three problems listed above in more detail and provide motivation for them.

1.1 Single Nucleotide Polymorphisms (SNPs)

The human genome can be considered as a long string over the four letter alphabet: $\{A,C,T,G\}$.¹ In total, the human genome has about 2.8 billion nucleotides packed into 23 chromosomes. Humans are *diploid* organisms and each individual has two copies of each chromosome (except for the X and Y chromosomes)². One copy is inherited from the mother and the other from the father. As DNA is transmitted from parent to child, many small and large scale mutations can take place in this sequence. Single base pair substitutions or point mutations substitute one nucleotide for another. For example, a single base pair mutation can change the DNA sequence “AACTGATAG” to “AACTCATAG”. If a mutation is in the reproductive cells of the parent, the offspring may inherit the mutation. Over many generations, this mutation may increase in frequency in a population of individuals, remain limited to a small number of individuals

¹ these correspond to the different nucleotides: A (adenine), C (cytosine), T (thymine), and G (guanine)

²males have the pair (X,Y) while females have two copies of X

or eventually die out. Mutations that increase in frequency such that both the original DNA sequence and the mutation become prevalent in a population (e.g. frequency $> 1\%$) are known as “polymorphisms”. Note that some mutations may completely replace the original DNA variant, e.g. if the mutation offers some form of selective advantage to an individual. Single Nucleotide Polymorphisms are point mutations that have become frequent in a population of individuals. Most SNPs seen in human populations are bi-allelic, i.e. there are two alleles seen in a population - the original nucleotide and the mutation. For a SNP to have three common alleles, a new mutation must happen at the same location in another individual and this mutation should also increase in frequency. The probability of observing this in human polymorphism data is low for two reasons: i) single base pair mutation rates in the human genome are low (of the order of 10^{-9} per base pair per generation), and ii) human populations are relatively recent in origin. For a bi-allelic SNPs with major allele ‘A’ and minor allele ‘a’, every individual can have one of three possible genotypes: two minor alleles (aa), two major alleles (AA) or both alleles (aA). When both chromosomes carry the same allele for a SNP, the individual is said to be homozygous. When the two chromosomes have different alleles, the individual is heterozygous.

1.1.1 SNPs and Haplotypes

Haplotypes refer to the sequence of alleles at a collection of SNPs on a single chromosome. In other words, a haplotype is the DNA sequence at the varying sites or SNPs. Figure 1.1.1 illustrates the concept of SNPs and haplotypes. Since humans are diploid, every individual has two haplotypes, e.g. the two haplotypes for the first individual in this region are *ACGTTC* and *AGGGAC*. The genotype for this individual can be represented as $A[C|G]G[T|G][T|A]C$. The process of determining the alleles that an individual carries at a SNP is called *genotyping*. Genotyping can determine whether the individual carries the minor allele on both chromosomes (aa), two major alleles (AA) or allele a on one chromosome and allele A on the other (aA/Aa). Given the genotype there

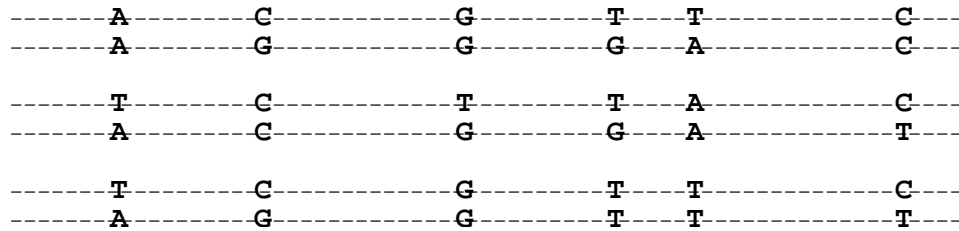


Figure 1.1: Illustration of the two copies from a region of a chromosome present in three individuals. Only the positions which are SNPs are shown, the letters in between are represented as '-'.

are many pairs of haplotypes that are consistent with it. For example there are 4 pairs of haplotypes that are consistent with the genotype for the first individual. Only one of these is the true pair of haplotypes. In the absence of molecular methods for determining haplotypes, haplotypes are inferred computationally from SNPs genotyped in a set of individuals from a population (Clark, 1990; Excoffier and Slatkin, 1995; Stephens et al., 2001; Niu et al., 2002; Stephens and Donnelly, 2003). This is known as *haplotype phasing* and there are a wide variety of methods based on different evolutionary models for obtaining haplotypes.

1.2 Meiotic Recombination

Mutation is the starting point of all genetic variation, however, there are other biological forces that can create genetic diversity. The most important among these is *meiotic recombination*. Recombination produces genetic diversity in a population by mixing of homologous chromosomes as they are passed on to the next generation. When DNA is passed onto the offspring from the parent, the two copies of each chromosome combine to produce a *mosaic* chromosome. This process where two chromosomes present in the parent are shuffled to produce a mosaic chromosome is called recombination (see Figure 1.2 for an illustration). It is this mosaic chromosome that is passed on to the offspring and can be different from the two chromosomes present in the parent. In evolutionary biology, recombination is believed to be an important mechanism for

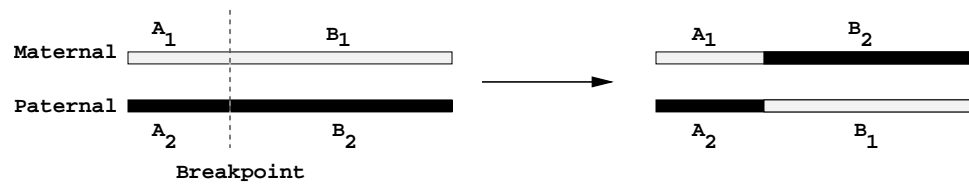


Figure 1.2: Illustration of a recombination event between the two chromosomes (maternal and paternal) in an individual. A recombination event produces two mosaic chromosomes, one which is passed on to the offspring.

producing new combination of genes thus allowing multiple beneficial variants which arose separately on different chromosomes to come together. There is great interest in understanding where recombination happens in the human genome, what genomic features determine its extent and how recombination rates change over time. This can provide insights into the evolutionary advantage provided by recombination in humans and in sexually reproducing species in general.

1.2.1 Recombination Rates

Genetic distance measures the amount of recombination between two markers (also referred to as loci) on a chromosome. Two loci are said to at a genetic distance of 1 M (Morgan) if the expected number of recombination events between them during a single meiosis event (formation of sperm/egg cell from two homologous chromosomes) is 1. The average recombination rate across the human genome is quite low: about 0.01 M per megabase or one recombination event per 100 Mb. In a large study, the average number of recombination events per generation was estimated to be 44 (females) and 27 (males) (Broman et al., 1998).

A *genetic map* is a sequence of ordered genetic markers³ with an estimate of the genetic distance between every pair of adjacent markers. Genetic maps are constructed by genotyping a large number of individuals that are closely related by a pedigree at the genetic markers and estimating the genetic distance by a simple count of the number of

³A genetic marker is a locus in the genome that is polymorphic in a population, i.e. has multiple alleles.

recombination events in each interval averaged over all meiosis events in the pedigree. Lot of variation has been observed in the distribution of recombination events across the human genome in genetic maps (Kong et al., 2002a). The resolution of genetic maps is limited to the megabase scale due to the low recombination rate in humans, i.e. genetic maps cannot reveal variation in recombination rates at the level of genes (1-10 kilobases). In order to detect a recombination event in a 1 kb interval of a chromosome, one will have to observe of the order of 10^4 meiosis events, which is experimentally infeasible.

There is great interest in understanding how fine-scale recombination rates vary across the human genome. Recombination hotspots are regions of the genome that have a very high rate of recombination compared to the background. Analyses of the distribution of recombination events in sperm DNA has revealed that recombination events cluster in small regions known as hotspots Kauppi et al. (2003); Jeffreys et al. (2005). However, these experimental techniques for estimating fine-scale recombination rates (recombination rates on the kilobase scale) in males (Jeffreys et al., 2000, 2001) are very laborious and limited to small regions (about few hundred kilobases) of the human genome. Methods for estimating fine-scale recombination rates from population data have been shown to be useful for detecting recombination hotspots McVean et al. (2004); Jeffreys et al. (2005).

1.2.2 Recombination and Linkage Disequilibrium (LD)

When markers are genotyped in a sample of unrelated individuals, one observes that the alleles at pairs of physically close markers show non-random correlation. In some cases, the alleles at multiple SNPs in segments of the genome are in perfect correlation. Why do we see such non-random associations at markers in populations of individuals who are not related to each other ? The answer lies in the shared ancestry of our chromosomes. A variant arises through a mutation event on a unique chromosome and shares a distinct combination of alleles at neighboring markers on this chromosome.

Individuals who inherit this variant will also tend to inherit the combination of alleles at markers physically close to the variant. Therefore, the allele will be associated with alleles at neighboring markers. Recombination will break down this association between the variant and alleles at neighboring markers by creating new combinations between alleles. However, one can still detect this association if the recombination rate is small and the number of generations since the variant arose is not large.

The haplotypes that we observe in a population of unrelated individuals are a result of recombination events (and other forces) that have happened over thousands of generations. In contrast to haplotypes of individuals related by a pedigree (typically over a few generations), we have information about many more recombination events but in a much weaker sense. We do not observe the underlying genealogy and hence cannot count the number of recombination events. Undeniably, there is some information about fine-scale recombination rates averaged over thousands of generations in this data. How do we extract this information from this data ? Consider two physically neighboring SNPs between which little or no recombination has happened in history of the population. One would expect the alleles at these two SNPs to be highly correlated. On the other hand, if the two SNPs are far apart and lot of recombination events have happened between them, one would expect no correlation between the alleles at the two SNPs. In population genetics, *Linkage Disequilibrium* quantifies this non-random correlation (or lack of) in alleles at neighboring markers in genotype data from unrelated individuals. Linkage disequilibrium (LD) is measured using pairwise association statistics such as D' and r^2 (Lewontin, 1964; Hedrick, 1987a; Pritchard and Przeworski, 2001). Consider two bi-allelic SNPs A and B with alleles A_1/A_2 and B_1/B_2 respectively. Let P_{A_i} be the frequency of allele A_i at locus A and similarly for locus B . Let $P_{A_1B_1}$ represent the frequency of the pair A_1B_1 and define $D = |P_{A_1B_1} - P_{A_1}P_{B_1}|$. Then

$$D' = \frac{D}{D_{max}}$$

where D_{max} is a normalizing factor so that D' lies between 0 and 1 independent of allele

frequencies. Another commonly used measure of LD, r^2 is defined as

$$\frac{D^2}{P_{A_1}P_{A_2}P_{B_1}P_{B_2}}$$

Significant LD is observed in human populations between markers at short distances (10-50 kilobases). This is primarily due to the low average recombination rate in humans (10^{-8} per base pair per generation) and the relatively recent origin of human populations. Greater LD is seen in European populations as compared to African populations which are older. LD is also greatly affected by other factors such as population history (migration and changes in population size), natural selection, etc. Nonetheless, LD has important implications for disease association mapping. Whole genome association studies genotype large number of cases (affected patients) and controls at a selected subset of SNPs distributed throughout the genome and identify SNPs that show statistically significant correlation with the disease phenotype. The rationale behind this approach is that even if the causal genetic variant is not genotyped, a SNP in LD with this variant is likely to show association with the disease phenotype. The HapMap project genotyped millions of SNPs in human populations so that LD information could be utilized for designing disease association studies, i.e for determining how many SNPs to type in a region to have enough power to detect disease associations. Next, we describe how we can detect historical recombination events using haplotype data from unrelated individuals.

1.3 Reconstructing Evolutionary Histories from Haplotype data

SNPs arise as a result of a mutation event (substitution of one base for another) that happened during human history. The commonly used *infinite sites assumption* in population genetics states that there is no recurrent mutation at a SNP. As there are only two alleles at every SNP, we can label the alleles as 0/1 and hence every haplotype

over m SNPs can be represented as a binary sequence of length m . Mathematically, a recombination event at column p , between two haplotypes A and B , produces a recombinant haplotype C , which is either a concatenation of $A[1 \dots p - 1]$ with $B[p \dots m]$ or $B[1 \dots p - 1]$ with $A[p \dots m]$.

In the absence of recombination (for example in the Y chromosome), each chromosome inherits some mutations from his parental chromosome and adds new ones that will be shared by all his descendants. Under the infinite-sites assumption, the history of the chromosomes can be explained by a *perfect phylogeny* (Gusfield, 1991). Informally, the perfect phylogeny tree derives the set of sequences starting from a root sequence through a sequence of mutations. The restriction that every location appears exactly once in the tree corresponds to the infinite-sites assumption, i.e. every polymorphic site mutates exactly once. A perfect phylogeny tree does not exist for any set of binary sequences. Gusfield (Gusfield, 1991) described an algorithm for finding a perfect phylogeny in time linear in the size of S (or proving that a perfect phylogeny does not exist). Recombination events cause genetic material to be inherited from two parental chromosomes and therefore the evolutionary history cannot be represented as a tree. In the presence of recombination, the evolutionary history can be represented through a directed graph known as the Ancestral Recombination Graph (Griffiths and Marjoram, 1996). In an ARG, a node with two incoming edges is called a *recombination* node. The two incoming edges to a recombination node are referred to as recombination edges. Nodes with only one incoming edge (a mutation edge) are called mutation nodes.

Formal Definition of Ancestral Recombination Graph Let S be a set of n binary sequences, of length m . An *Ancestral Recombination Graph* G for S is a directed acyclic (no directed cycles) graph with root R and the following properties:

1. Each node \mathcal{N} of G corresponds to a binary sequence of length m (denoted by $\text{seq}(\mathcal{N})$). Each sequence in S corresponds to a leaf in G .
2. Each *mutation* edge in G corresponds to a set of sites.

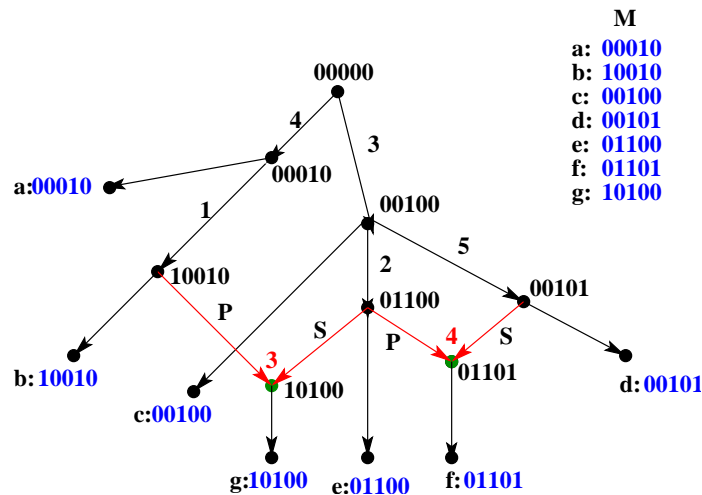


Figure 1.3: Example of an Ancestral Recombination Graph explaining a set S of binary sequences with two recombination nodes (denoted by green circles). The labels of the external nodes (leaves) are shown in blue. The incoming edges for the two recombination nodes are colored red. The recombinant sequence 10100 (d) is derived from a recombination event between 10010 (\mathcal{P}) and 01100 (\mathcal{S}) where the first two columns of d come from \mathcal{P} and the last three columns from \mathcal{S} . Hence the recombination node is labeled 3.

3. If nodes A and B are connected by an mutation edge, then the sequences $\text{seq}(A)$ and $\text{seq}(B)$ differ at exactly the sites corresponding to the edge.
4. Each recombination node v is associated with an integer r_v , $2 \leq r_v \leq m$, called the recombination point for v . Let \mathcal{P} and \mathcal{S} be the binary sequences corresponding to the two “parent” nodes connected to v by recombination edges. The sequence corresponding to node v is a concatenation of the first $r_v - 1$ elements of \mathcal{P} with the last $m - r_v + 1$ elements of \mathcal{S} .

An ancestral recombination graph G that explains a set S of n binary sequences represents a *possible* evolutionary history of the sequences under the assumption that the sequences were derived from an ancestral sequence (the root) through a sequence of mutation and recombination events. Note that there could be multiple ARGs explaining the same set of sequences each with different number of recombination nodes. Figure 1.3 illustrates the structure of an ARG (example from (D.Gusfield et al., 2003)).

1.3.1 Counting Recombination Events

Haplotype data obtained from a random sample of individuals from a population contains information about recombination events that have happened over thousands of generations in the history of that population. However, reconstructing the evolutionary history of a set of sequences that have undergone recombination and mutation is difficult since many recombination events do not create new sequences. A parsimonious approach is to estimate the minimum number of recombination events required to explain the sample of sequences. We define R_{min} to be the minimum number of recombinations required to explain S , i.e. there exists an ARG with R_{min} number of recombination nodes which explains S and there is no ARG with fewer recombination nodes that explains S . Reconstructing the ARG with the minimum number of recombination events is a challenging computational problem. This problem is computationally hard and lacks even an exponential time algorithm. Hein (Hein, 1990, 1993) proposed methods for reconstructing parsimonious ARGs, however the complexity of these methods is super-exponential and the methods practical for about 8-10 sequences. This problem was recently revisited by Gusfield and colleagues who gave polynomial time algorithms for reconstructing minimal ARGs which have the property that every recombination node is in its own edge disjoint cycle (D.Gusfield et al., 2003, 2004). Since one is interested in the number R_{min} rather than the actual evolutionary histories, research in this area has focused on computing lower bounds on R_{min} . This can be achieved without explicitly considering evolutionary histories and is a somewhat simpler problem.

Consider two SNPs A and B with 0 representing the ancestral allele and 1 the derived allele at each SNP. There are four possible pairs of sequences $\{00, 01, 10, 11\}$ that one can observe at these two SNPs. Any sequence of two mutations starting from the ancestral sequence 00 can produce at most 3 different pairs, one example is $\{00, 01, 10\}$. The fourth pair 11, can be produced by a recombination event between the two sequences 10 and 01. Therefore, if we observe all four pairs at the two SNPs, one can infer that at least one recombination event happened between the two sites at some time

in the history of the population (assuming that no site mutated more than once). This is known as the *four-gamete test* in population genetics. The first lower bound on the minimum number of recombination events was proposed by Hudson and Kaplan (1985) using the four-gamete test. Significantly better lower bounds were proposed by Myers and Griffiths (2003) using haplotype diversity and an indirect notion of evolutionary histories. In Chapter 2, we mathematically describe the problem of computing lower bounds on R_{min} and analyze the computational complexity of the two lower bounds of Myers and Griffiths (2003). We propose faster methods for computing these lower bounds and also introduce a new lower bound that is provably better than all previous lower bounds. We show that these methods can detect higher number of recombination events for a haplotype dataset from a region in the lipoprotein lipase gene than previous lower bounds. We apply our methods to two datasets for which *recombination hotspots* have been experimentally determined and demonstrate a high density of detectable recombination events in the regions annotated as recombination hotspots.

In Chapter 3 we explore the problem of computing lower bounds on R_{min} using conflicts between pairs of sites. We analyze the conflict graph for a set of sequences and demonstrate that number of non-trivial connected components in the conflict graph is a lower bound on the minimum number of recombination events required to explain the set of sequences under the infinite sites model of evolution. We show that in many cases, this lower bound, R_c is a better bound than the haplotype lower bound R_h . Our results also offer some insight into the structural properties of this graph and are of interest for the general ARG reconstruction problem.

1.4 Finding Inversions using SNP haplotype data

Inversions represent a type of structural variation in which a genomic segment is reversed or inverted. Inversion polymorphisms are well known in the genus *Drosophila* but less is known about the extent and frequency of inversions in the human genome. Unlike deletions which cause miscalled genotypes and can lead to Mendelian inconsis-

tencies (McCarroll et al., 2006; Conrad et al., 2006), inversions are copy neutral and do not affect the SNP genotypes. However, large polymorphic inversions can affect haplotype patterns in a population by suppressing recombination in the inverted region for individuals who are heterozygous for the inversion, i.e. carry both the non-inverted and the inverted allele. Lack of recombination between the two orientations also causes them to evolve independently accumulating mutations that are specific to each orientation. We use an indirect approach to detect large inversions for which the inverted allele (with respect to the reference sequence assembly) represents the major allele. As described earlier, Linkage Disequilibrium measures the correlation in alleles at two markers and tends to decay with increasing genetic distance between the markers. Two SNPs that are physically close tend to show higher levels of LD than SNPs that are physically distant. Our method tries to identify pairs of breakpoints for which the distant LD is unusually strong while the LD between SNPs across the breakpoint is low. This type of LD pattern is indicative of the fact that the region is actually inverted in a large fraction of the chromosomes in a population. In chapter 4, we describe a statistical method for detecting large inversions in the human genome based on this unusual LD pattern. We apply our method to haplotype data from the International HapMap project to generate a list of candidate inversions and obtain additional evidence supporting these predictions through genomic analysis. Our method is applicable to population data from other organisms and represents an useful technique for scanning for large inversions in the absence of sequencing data.

1.5 The Haplotype Assembly Problem

Humans are diploid organisms with two copies of each chromosome (except the sex chromosomes). The two chromosomes are homologous and differ at a number of sites, a large fraction of which correspond to single base-pair differences commonly known as Single Nucleotide Polymorphisms (SNPs). The genome sequence assembly for a chromosome is an arbitrary mix of the two haploid chromosomes and does not

contain information about which alleles are present on the same chromosome. The two *haplotypes* (described by the combination of alleles at variant sites on a single chromosome) represent the complete information on DNA variation in an individual. SNP chips can now interrogate up to a million SNPs in each individual, making the genotyping effort cost-effective. However, haplotype information remains difficult to obtain experimentally. Haplotypes are typically inferred from population SNP data using “haplotype phasing” algorithms (Gusfield, 2002; Stephens et al., 2001; Bafna et al., 2003; Eskin et al., 2003). These algorithms exploit Linkage Disequilibrium (LD); the correlation between alleles at neighboring SNPs in a population to reconstruct haplotypes. The great variation in recombination rates and Linkage Disequilibrium across the human genome limits the accuracy of these methods.

An alternative way to obtain haplotypic information is to reconstruct the two haplotypes for an individual using DNA sequence fragments. A sequenced fragment is a piece of one of the chromosomes and a fragment that is long enough will cover multiple variant sites and provide information about the alleles at those sites present on a single chromosome. If a large fraction of the fragments are long enough to encompass multiple variant sites, and the shotgun sequencing has sufficient coverage to provide overlaps between fragments, the fragments can be assembled to reconstruct long haplotypes. The haplotype assembly problem, also known as the Single Individual Haplotyping problem, was introduced in the context of Single Nucleotide Polymorphisms (SNPs) by Lancia et al. (2001) who described three optimization formulations for solving this problem. The problem has been shown to be computationally hard under various combinatorial objective functions (Lancia et al., 2001; Bafna et al., 2005; Cilibrasi et al., 2005) (e.g. Minimum Fragment Removal (MFR), Minimum Error Correction (MEC), Minimum SNP Removal (MSR)). Efficient algorithms exist for optimizing the MFR objective when all fragments are gap-less (Rizzi et al., 2002; Lippert et al., 2002). However, the lack of real sequencing data has limited the development and evaluation of computational methods for this problem. Recently, Levy and colleagues announced the diploid genome sequence of a single human individual (Levy et al., 2007), referred to as

HuRef. They also demonstrated that the quality of the data and the presence of paired-end reads makes haplotype assembly feasible. A simple greedy heuristic was used for reconstructing haplotypes. The heuristic works well for the *HuRef* sequence data, but results indicate that it can be improved.

The final part of this thesis is devoted to the haplotype assembly problem. In chapter 5, we describe a Markov chain Monte Carlo algorithm for this problem that is based on the notion of computing certain cuts in graphs derived from the sequenced fragments. This algorithm was motivated by the availability of the *HuRef* sequence data. In chapter 6, using similar ideas, we describe a combinatorial algorithm for this problem that tries to optimize a certain error measure of the haplotype assembly. In chapter 7, we investigate the mixing properties of the Markov chain underlying our algorithm and show that a cut-based Markov chain has polynomial mixing time for a family of fragment matrices.

Chapter 2

Counting Recombination Events: Lower bounds and Recombination Hotspots

Meiotic recombination is a major mechanism responsible for creating genetic diversity in many species. Although all genetic variation starts from mutation, recombination can give rise to new variants by combining types already present in the population. Recombination events break up haplotypes as they are passed from one generation to the next during gametogenesis and greatly influence the patterns of haplotype variation in human population data. Until recently, the variation in recombination rates on a genome-wide scale was primarily studied by genotyping large number of individuals related by a pedigree, and estimating the recombination rates between the genotyped markers by a direct count of recombination events. However, constructing such genetic maps (Kong et al., 2002b) requires high marker density and can only provide information about variation in recombination rates on the mega-base scale. In contrast to genotype data from families, population genetic data (genotype data from unrelated individuals) contains information about recombination events accumulated over many generations and can reveal fine-scale variation in recombination rates on the kilo-base

scale. In the post-genomic era, the emergence of genome-wide diversity studies, such as the HapMap project (The International HapMap Consortium, 2003), has enabled the characterization of fine-scale distribution of recombination events across the genome. Initial analyses of human polymorphism data (Gabriel et al., 2002; Daly et al., 2001; Jeffreys et al., 2000) suggested an interesting block like structure of the genome, where long stretches known as *LD blocks* (with little or no diversity) show signs of little or no recombination and the recombination events cluster in so called *recombination hotspots*. To enable a more quantitative analysis of these datasets, a variety of statistical methods based on different population genetics models have been proposed to estimate recombination rates from genotype data (see e.g. (Fearnhead and Donnelly, 2001; McVean et al., 2002; Hudson, 2001; Li and Stephens, 2003)). Sperm typing is an experimental technique that can reveal fine scale variation in recombination rates by counting crossover events from sperm DNA samples. Sperm crossover analysis from two regions from the human genome (Jeffreys et al., 2001, 2005) identified several short (1-2KB) regions with elevated crossover rates. Most of these crossover hotspots were also detected using coalescent based computational methods (Li and Stephens, 2003; McVean et al., 2004; Fearnhead et al., 2004; Stephens and Donnelly, 2003) with some differences between the recombination rates estimated from the two methods.

In genotype data from individuals related by a pedigree, it is possible to obtain a estimate of the number of the recombination events between every pair of markers. In the absence of any genealogical information about the genotyped individuals, obtaining a direct count is not possible. Some coalescent based approaches (Fearnhead and Donnelly, 2001) estimate recombination rates by integrating over large number of genealogies consistent with the observed data. In contrast to explicitly modeling the evolutionary history of chromosomes to infer recombination rates, an alternative approach for characterizing the variation in recombination is to obtain a count of obligate recombination events. Population genetic data, in particular haplotype data contains signature patterns left behind by historical recombination events.

A parsimonious approach to inferring recombination events from haplotypes is

to compute the minimum number of recombination events required to construct an evolutionary history of the sample assuming that each segregating site mutates only once. This problem is computationally challenging and has resisted efforts for even an exponential time algorithm (Hein, 1990, 1993; Song and Hein, 2003; Wang et al., 2001a; D.Gusfield et al., 2003). Therefore, research in this area has focused on computing lower bounds on the minimum number of recombination events. Although most historical recombination events leave no imprint in the data, one expects that regions with elevated recombination rates will have a large number of detectable recombination events in comparison to surrounding regions.

For almost two decades, the R_M lower bound (Hudson and Kaplan, 1985) has been used to detect the presence of recombination in haplotype data (see e.g. (Wang et al., 2001b)). Recently, the R_h lower bound (Myers and Griffiths, 2003) was demonstrated to be much more powerful than the R_M lower bound for detecting recombination events through simulation studies and detected a strong clustering of recombination events in the center of the lipoprotein lipase gene (Nickerson et al., 1998). This region has previously been characterized to be a putative recombination hotspot (Templeton et al., 2000). The R_h lower bound was applied to detect recombination events in the β -globin gene cluster (Fearnhead et al., 2004) which has a well-characterized recombination hotspot. It was reported that the results obtained using the lower bound were consistent with the estimates obtained using a full likelihood method.

Outline of chapter In this chapter, our objective is to explore the problem of computing lower bounds on the number of recombination events. We present many results on the computational complexity of computing recombination lower bounds and their application to real haplotype datasets to reveal fine-scale distribution of recombination events.

- We provide a theoretical formulation for the lower bound R_h and show that it is NP-hard to compute this bound. However, on the positive side, using the greedy algorithm for the set cover problem (Johnson, 1972), we present a $O(mn^2)$ time

algorithm which computes a lower bound R_g for a dataset with n rows and m segregating sites. Using simulations under the coalescent, we show that this new lower bound is faster than the Recmin program¹. and is more sensitive than the Recmin bound to changes in the recombination rate, especially for higher recombination rates.

- Most real haplotype datasets have some amount of missing data. A simple way of handling missing data is to not consider markers which have missing alleles for some haplotypes. We extend the lower bound R_g to compute a lower bound utilizing information from all markers in the presence of missing data. These bounds applied to the LPL dataset (Nickerson et al., 1998) detect many more recombination events (in comparison to the number detected by ignoring the sites with missing data) which provide stronger support for the presence of a recombination hotspot (Templeton et al., 2000).
- We apply our methods to genotype data from two long regions (several hundred kilobases) from the human genome and show that these can indicate the presence of most of the recombination hotspots that were detected experimentally using sperm typing (Jeffreys et al., 2001, 2005).
- We give an $O(m2^n)$ time algorithm for computing R_s which enables us to apply it to real datasets. The previous implementation (Myers and Griffiths, 2003) had only an $\Omega(m \cdot n!)$ bound and is intractable for more than 10-15 haplotypes.
- We show that the lower bound R_s can underestimate the true number of recombination events since it does not consider missing haplotypes. We propose a new bound R_I which extends R_s using the notion of intermediate haplotypes. The R_I bound for the haplotypes from the ADH locus of *Drosophila Melanogaster* (Kreitman, 1983) is 7 which is optimal for this dataset (equal to

¹ Recmin is the program that implements the R_h lower bound (Myers and Griffiths, 2003). Throughout this chapter, we will refer to the bound computed using Recmin (enhance =0) as the Recmin lower bound

the upper bound of 7). We also show that the R_I bound is better than all previous bounds on several datasets from the SeattleSNP database (SeattleSNPs, <http://pga.gs.washington.edu>, March 2004).

2.1 Preliminaries

2.1.1 Lower Bounds on the Minimum Number of Recombination Events

The lower bound R_M (Hudson and Kaplan, 1985) is based on the *four-gamete test*; if for a pair of SNP's with ancestral and mutant alleles a/b and c/d respectively, all four possible gametes (ac, ad, bc, bd) are present, then at least one recombination event must have happened between the pair of loci under the assumption that no site mutates more than once. Based on this idea, one can find all intervals in which recombination must have occurred and choose the largest set of non-overlapping intervals from this collection. The bound R_M is the number of intervals in this set. However, R_M is a conservative estimate of the actual number of recombination events (Hudson and Kaplan, 1985). One can use haplotype diversity to infer more than one recombination event in an interval. Consider an interval with m segregating sites. If $n(> m + 1)$ distinct haplotypes are observed in this interval, then at most m haplotypes can be explained using mutation events. Assuming that the ancestral haplotype is present in the sample, the remaining $n - m - 1$ haplotypes must arise due to recombination events. Hence, one can infer a lower bound of $n - m - 1$ for the interval. Moreover, one can choose any subset of segregating sites for an interval and compute this difference to obtain another lower bound for that region. Taking the maximum bound over all subsets of segregating sites in a particular region gives the best lower bound, denoted as R_h (Myers and Griffiths, 2003).

The bounds R_M and R_h do not explicitly consider possible histories of the sample. The lower bound R_s (Myers and Griffiths, 2003), computes for every history (an

ordering of the haplotypes), a simplified number of recombination events, such that any phylogenetic network that is consistent with this history, requires more recombination events than this number. By minimizing over all possible histories, one obtains a lower bound on the minimum number of recombination events. The algorithm for computing R_s (for a precise description see (Myers and Griffiths, 2003)) performs three kinds of operations on a given matrix: row deletion, column deletion and non-redundant row removal. A *row deletion* can be performed if the given row is identical to another row in the matrix. Such a row is also referred to as a *redundant* row. A *column deletion* can be done if the column (site) is *non-informative* (all but one rows have the same allele at this site). A *non-redundant row removal* is a row removal when there are no non-informative sites in the matrix and no redundant rows. Given an ordering of the n rows, the algorithm performs a sequence of column deletions, row deletions and non-redundant row removals until there is no row left in the matrix M . The minimum number of non-redundant row removal events over all possible histories gives the bound R_s . Since, the procedure considers all $n!$ histories, the worst case complexity of this procedure is $\Omega(m.n!)$.

2.1.2 Combining Local Recombination Bounds

Myers and Griffiths (2003) presented a general framework for computing recombination lower bounds from haplotype data. This framework can combine local recombination bounds on continuous subregions of a larger region to obtain recombination bounds for the larger parent region. Consider a matrix M with m segregating sites labeled 1 to m . Suppose that one has computed, for every interval (i, j) ($1 \leq i < j \leq m$), a lower bound b_{ij} on the number of recombination events between the sites i and j . Each local lower bound b_{ij} can be computed by any lower bound method described previously and bounds for different intervals may be obtained by different methods.

In the second step, which is essentially a dynamic programming algorithm, one computes a new lower bound B_{ij} on the minimum number of recombination events

between the sites i and j using the local bounds $b_{i'j'}, i' \leq i < j \leq j'$. The local bound B_{ij} can be computed as $B_{ij} = \max_{k=i+1}^{j-1} (B_{ik} + b_{kj})$. Note that the combined lower bound B_{ij} can be substantially better than the corresponding local bound b_{ij} for an interval (i, j) . It is important to note that all the practical results are obtained by computing lower bounds (by using the corresponding lower bound method) for all intervals of length w (specified as a parameter) for the given dataset, and combining them using the dynamic programming algorithm.

2.2 Bounds based on Haplotype Diversity

Consider a matrix M and let $S' \subseteq S$ be a subset of sites in M . For a subset S' of segregating sites, we denote the set of distinct haplotypes induced by S' as $H(S')$. The R_h bound (Myers and Griffiths, 2003) is based on the observation that $|H(S')| - |S'| - 1$ is a lower bound on the number of recombinations for every subset S' . Since the number of subsets is 2^w for a region of width w , The Recmin program (Myers and Griffiths, 2003) use the approach of computing this difference for subsets of size at most s where $s < w$ is a specified parameter. Increasing s can provide better bounds with an increase in computation time since the running time is exponential in s . We define the algorithmic problem associated with the computation of the bound R_h as follows:

MDS: Most Discriminative SNP subset problem

Input: A binary matrix M and an integer k .

Output: Is there a subset S' of S , where S is the set of columns of M , such that $|H(S')| - |S'| - 1 \geq k$.

Computing the R_h bound is equivalent to finding the largest value of k for which the MDS problem has a solution. We show that MDS problem is NP-complete by using a reduction from the *Test Collection Problem* (Garey and Johnson, 1979). An instance of the test collection problem (TCS for short) consists of a collection \mathcal{C} of subsets of a finite set S and an integer k , and the objective is to decide if there is a sub-collection $\mathcal{C}' \subseteq \mathcal{C}$ such that for each $x, y \in S$ there exists $c \in \mathcal{C}'$ that contains exactly one of x

and y and $|\mathcal{C}'| \leq k$. An instance of the test collection problem can be encoded as a binary matrix M of size $|\mathcal{S}| \times |\mathcal{C}|$. Each row of the matrix corresponds to an element of the finite set \mathcal{S} and $M[x, c] = 1$ if the subset c contains the element x and 0 otherwise. Here, the objective is to find a subset S' of the columns of M of size at most k such that for every pair of rows in M , there is a column in S' that can distinguish between them, i.e. $|S'| \leq k$ and $H(S') = |\mathcal{S}|$. Using this encoding we show that the MDS problem is NP-complete.

Lemma 1: The MDS problem is NP-complete.

Proof: We prove the k -TCS and the $(n - k - 1)$ -MDS problems to be equivalent. Consider a subset S' of S such that $|H(S')| = n$ and $|S'| \leq k$, i.e. S' is a valid solution of the k -TCS problem. It follows that for the subset S' , $|H(S')| - |S'| - 1 \geq n - k - 1$. Therefore, S' is a valid solution for the $(n - k - 1)$ -MDS problem.

Now, let $S' \subseteq S$ be a solution of the $(n - k - 1)$ -MDS problem, i.e. $|H(S')| - |S'| - 1 \geq n - k - 1$. Consider a haplotype $h \in H$ such that $h \notin H(S')$. For such a haplotype, there is exactly one haplotype h' in $H(S')$ that is identical to h , since all haplotypes in $H(S')$ are distinct. Also, there is a site $s \in S - S'$ such that the character at this site in h is different from the character at this site in haplotype h' . Hence, we can add the site s to S' and the haplotype h to $H(S')$ to get another set S'' such that $|S''| = |S'| + 1$ and $|H(S'')| = |H(S')| + 1$. Clearly, S'' is also a solution for the $(n - k - 1)$ -MDS problem. Inductively, we can add all haplotypes not present in $H(S')$ to obtain a subset of sites S^* such that $H(S^*) = H$. Therefore, $|S^*| \leq |H(S^*)| - (n - k) = k$. Hence S^* is a solution for the k -TCS problem.

Therefore, the k -TCS problem and the $(n - k - 1)$ -MDS problem are equivalent. The NP-completeness of the MDS problem follows. \clubsuit

2.2.1 The lower bound R_g

From the encoding for the MDS problem, it is easy to see that computing the bound R_h is equivalent to finding a smallest subset of columns C such that for every

pair of rows (x, y) , there is at least one column $c \in C$ such that $M[x, c] \neq M[y, c]$. We adapt the standard greedy algorithm for the set cover problem (Johnson, 1972) to devise an algorithm for computing a lower bound; denoted as R_g . It is well known that the greedy algorithm gives a $(1 + 2 \ln n)$ approximation for the test collection problem where $n = |\mathcal{S}|$, the size of the ground set. However, this approximation ratio does not apply to the MDS problem.

Algorithm for computing the lower bound R_g :

1. If two rows in M are identical, coalesce them. If a column s is non-informative, remove the column s . Repeat while it is possible to perform one of these operations.
2. Let M' be the reduced matrix with n rows and m sites
3. Initialize $I = \phi$ and $d(x, y) = 0$ for all $x, y \in M'$
4. **while** $d(x, y) = 0$ for some pair of rows
 5. Let s' be the column for which $\sum_{(x,y)} (1 - d(x, y)) \wedge (M'[x, s'] \oplus M'[y, s'])$ is maximum
 6. set $d(x, y) = 1$ for all (x, y) s.t. $M'[x, s'] \neq M'[y, s']$
 7. $I = I \cup \{s'\}$
 8. **end while**
9. Return $|H(I)| - |I| - 1$

2.2.2 Comparison of the lower bound R_g with previous bounds

In order to compare the new lower bound R_g against previous bounds, we use simulated data generated under the coalescent (Kingman, 1982; Hudson, 1990; Rosenberg and Nordborg, 2002) using the MS program (Hudson, 2002). The coalescent is a

Table 2.1: Properties of the three lower bounds: R_g , Recmin program (Myers and Griffiths, 2003) and R_M (Hudson and Kaplan, 1985) and two other summary statistics for samples of size $n = 100$ and $\theta = 10$ and a region of length 10KB. Here H is the number of distinct haplotypes and R is the actual number of recombination events in the genealogy of the sample (generated using the MS program). Each point was obtained using 10000 samples. Recmin program was run with the default settings ($w=12$ and $s=5$). The coefficient of variation (denoted by c.v.) for a statistic is the standard deviation divided by the mean for that statistic.

Statistics	Recombination rate							
	1	2	5	10	20	25	50	100
H Mean	25.27	26.09	28.35	31.84	37.80	40.53	50.84	63.60
H c.v.	0.16	0.15	0.15	0.14	0.13	0.12	0.11	0.10
R_g Mean	1.23	2.27	5.09	9.03	15.80	18.94	31.54	49.69
R_g c.v.	0.89	0.65	0.44	0.34	0.27	0.25	0.21	0.19
R_M Mean	1.02	1.68	3.03	4.44	6.29	7.55	9.39	12.07
R_M c.v.	0.83	0.61	0.44	0.37	0.31	0.29	0.25	0.23
Recmin Mean	1.23	2.29	4.88	8.26	13.58	15.90	24.86	36.80
Recmin c.v.	0.88	0.63	0.43	0.34	0.28	0.26	0.23	0.22
R Mean	5.21	10.49	27.19	57.49	126.45	165.58	388.76	966.77
R c.v.	0.49	0.38	0.29	0.23	0.20	0.19	0.15	0.13
R_g (secs)	37	47	82	173	391	515	1150	2449
Recmin (secs)	34	59	168	410	880	1068	1880	3160

standard framework for simulating population genetic data. Under the coalescent, the history of a sample of n sequences is a stochastic process governed by two parameters: the scaled mutation rate $\theta = 4N\mu$ and the recombination rate $\rho = 4Nr$. Here, N is the effective population size, and μ and r are the per generation mutation rate and recombination rate for the whole region respectively. We simulated data under a neutral model with no population structure, constant population size and assuming the infinite sites model for mutations. Most simulations were done with a sample size of 100 and mutation rate in the range 1-2 per kb.

In table 2.1, we compare the mean and the coefficient of variation of various lower bounds with recombination rate varying from 0.1 per kb to 10 per kb. It is clear that our new lower bound R_g is more sensitive than the Recmin lower bound (using

default parameters) to changes in recombination rate, especially for higher recombination rates. Furthermore, the time to compute the bound R_g is always less than that for the Recmin program (except for $\rho = 0.1/kb$). Note that the Recmin lower bound will increase as one increases the parameters w and s and eventually will be at least as good as the R_g bound. To see how the performance of Recmin changes as we increase the parameters, we ran Recmin for 100000 samples generated with $\theta = 10$ and $\rho = 50$ with three different parameter settings. For the default settings of $w = 12$ and $s = 5$, the mean Recmin bound is 24.86 computed in about 30 minutes. Increasing w to 15 and s to 8, the mean increases to 27.31 but the running time doubles. With parameters $w = 20$ and $s = 10$, the mean increases to 30.03 but the program takes more than 25 hours to complete. In contrast, the R_g lower bound with a window size of 30 returns a mean of 31.54 in less than 20 minutes. The running time of Recmin is proportional to $\sum_{i=2}^s \binom{w}{i}$ where w is the maximum number of segregating sites in a region for which the local bound is computed and s is the maximum subset size used for computing the bound. In comparison, in order to compute the best bound by combining the local R_g bounds, we require only one parameter, i.e maximum width and the overall running time is $O(n^2mw^2)$.

Note that since the R_g lower bound is computed using a greedy procedure and is not guaranteed to be optimal, it may be worse than the Recmin lower bound for an individual dataset. However, comparison of the lower bound R_g with the Recmin bound for individual datasets revealed that the lower bound R_g was rarely worse than the corresponding Recmin bound. Moreover, it is difficult to decide what values of s and w will find the best Recmin bound. Choosing large values will result in prohibitive running times (as demonstrated above), while the bound with the default settings becomes increasingly sub-optimal as the recombination rate increases. Although Recmin can be used to compute the best bound for an individual sample, in order to empirically estimate the properties of the haplotype lower bound and its sensitivity to the recombination rate, it is important to have a fast method for computing the lower bound. In this respect, the R_g bound has the advantage that it can compute a bound equal to (or close

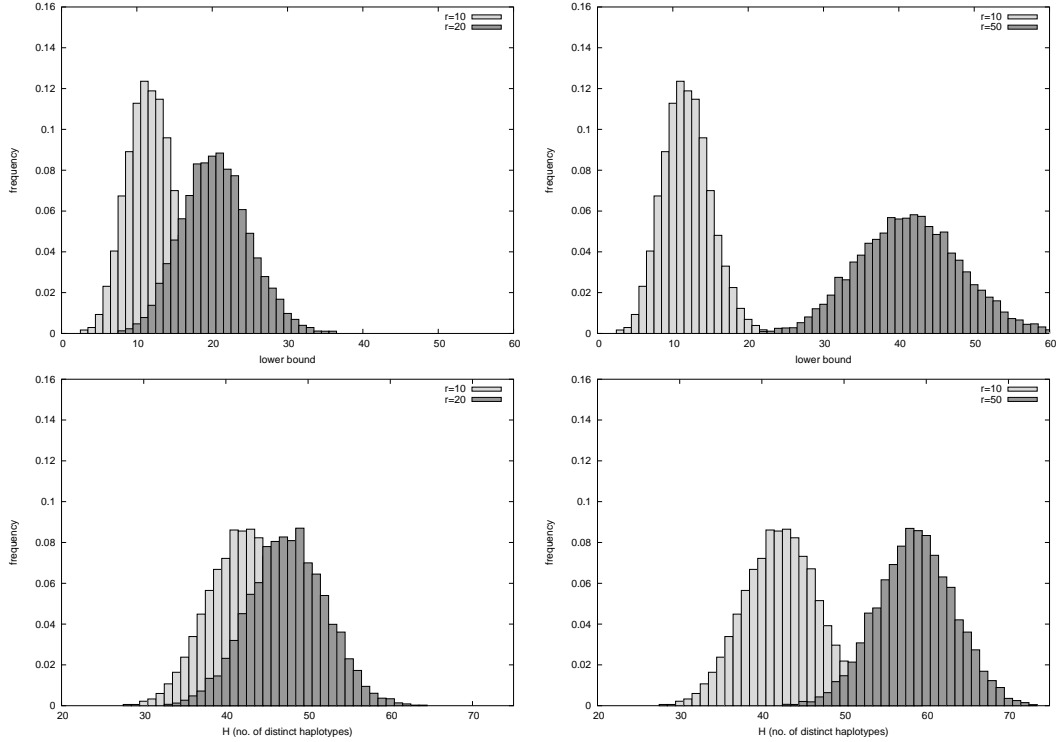


Figure 2.1: Distribution of the R_g lower bound and the number of distinct haplotypes for different recombination rates. On the left are frequency histograms of R_g and H for $\theta = 20$ and two different ρ values (10 and 20). The histograms on the right are for $\theta = 20$ and two different ρ values (10 and 50). The plots were generated using 10000 samples each.

to) the optimal R_h bound for a large range of recombination rates.

The number of distinct haplotypes H in a sample and the lower bound R_M have been shown to be good summary statistics for estimating the recombination rate from a sample of haplotypes using coalescent simulations (Wall, 2000). The estimated mean value of the lower bounds R_h and R_g show that these lower bounds are much more sensitive to changes in recombination rate than R_M . It is an interesting question as to whether these bounds could be better recombination summary statistics for a sample. In figure 2.1, we plot the distribution of the two summary statistics, R_g and H for two different values of ρ . In the first plots on the left, we compare the histograms for $\rho = 10$ and $\rho = 20$. From the plots, it is apparent that the distribution of the lower bound

shifts towards the right as the recombination rate is doubled and furthermore there is almost no overlap between the two distributions with a recombination rate difference of five-fold. Looking at the corresponding plots for H , we observe that R_g seems to be more sensitive to changes in recombination rate, although the distribution for H has a smaller spread. In general, the probability distribution of the R_h lower bound or any other summary statistic such as H is a function of the two variables θ and ρ . A summary statistic X is expected to be a good estimator of the recombination rate if the random variable $(X, \rho|\theta)$ is sensitive to changes in the recombination rate ρ and has a small coefficient of variation for a fixed ρ . Preliminary results indicate that the lower bound is a good summary statistic for estimating recombination rates after correcting for the variation in the number of segregating sites. (Bafna and Bansal, unpublished results)

2.2.3 Bounds for Haplotypes with Missing Data

A complete haplotype is an element of $\{0, 1\}^m$ where m is the number of SNP's and the j -th component indicates the allele at that position. However, due to genotyping errors or other reasons, the genotype at a particular position for a individual is sometimes undetermined. In such a scenario, some of the haplotypes are partial or incomplete. A partial haplotype is an element of $\{0, 1, ?\}^m$ where $?$ represents the positions where the allele is unknown. Since most real haplotype datasets have some amount of missing data, it is important to find efficient methods for computing recombination lower bounds for haplotypes with missing data. We show how the greedy algorithm for computing R_g can be extended to handle haplotypes with missing data without much increase in the computational complexity. We first need to modify the definition of a non-informative site. A site is defined to be non-informative if it has all but one alleles of one type (ignoring the missing alleles). With this modified definition, the algorithm for computing R_g described in section 2.2.1 remains unchanged for the first 8 steps. Recall that in the last step of the algorithm, we return the bound $H(I) - I - 1$. For a matrix with missing entries, it is not straightforward to compute $H(I)$. However, consider an assignment to the

?'s that minimizes $H(I)$. Then the difference $H(I) - I - 1$ gives a valid lower bound, i.e. a bound which is valid for all possible assignments to the missing entries. However, for minimizing $H(I)$ one has to solve the minimum haplotype completion problem; where given an haplotype matrix with missing entries, the objective is to complete the missing entries so as to minimize the number of distinct haplotypes. This problem was shown to be NP-hard (Kimmel and Shamir, 2004). To get a lower bound on the number of distinct haplotypes induced by the given data, we construct the compatibility graph on the set of haplotypes, where two haplotypes are connected by an edge if they are identical (treating the missing entries as don't cares). The number of connected components in this graph gives a valid lower bound on the minimum number of distinct haplotypes.

2.2.4 Application to Haplotype Data from LPL locus

A 9.7-kb region from the human LPL gene was sequenced by Nickerson. et al. (Nickerson et al., 1998) in 71 individuals from three different populations. The haplotype data comprised of 88 haplotypes defined by 69 variable sites with about 1.2% missing data. This data has previously been analyzed for haplotype diversity and recombination (Clark et al., 1998; Templeton et al., 2000; Myers and Griffiths, 2003). In table 2.2, we compare the bounds obtained for different sub-regions of the LPL region for various populations. The overall bound for the whole region is 70 if one ignores the sites with missing data (see (Myers and Griffiths, 2003)), while our method for computing lower bounds including missing data detects 87 recombination events. Templeton et al. (Templeton et al., 2000) had found that the 29 recombination events detected using their method to be clustered near the center of the region (approximately between the sites 2987 and 4872). It is interesting to note that number of detected recombination events (37) in this region increases significantly (from 22) when one takes into account the sites with missing alleles. Thus, the bounds obtained using our improved methods which can handle missing data, seem to provide strong support for the presence of a recombination hotspot suggested by Templeton et al. (Templeton et al., 2000). This

Table 2.2: The number of detected recombination events using bounds for missing data for the LPL datasets. The number in bracket indicates the density of detected recombination events per kb. The middle region (2987-4872) corresponds to the suggested hotspot (Templeton et al., 2000).

Region	Site Range			
	106-2987	2987-4872	4872-9721	Full
Jackson	10(3.47)	11(5.84)	17(3.51)	39(4.06)
Finland	2(0.69)	13(6.90)	13(2.68)	31(3.22)
Rochester	1(0.35)	13(6.90)	7(1.44)	22(2.89)
Combined	13(4.51)	37(19.63)	36(7.42)	87(9.05)

demonstrates that the ability to extract past recombination events can be crucial to detecting regions with elevated recombination rates.

2.2.5 Application of Lower Bounds to reveal Recombination Hotspots

In humans, individual hotspots have been identified using pedigree studies and sperm crossover analysis. However, characterizing fine-scale variation in recombination rates using pedigree studies (at the kb scale) is difficult and performing sperm analyses on a genome-wide scale is experimentally infeasible. Recombination hotspots are defined as regions in which the crossover rates are significantly larger than the rates in the surrounding regions. Detecting hotspots is important for disease association studies and understanding the biological mechanisms behind the origin and evolution of hotspots (for some recent work see (Ptak et al., 2004, 2005; Winckler et al., 2005)).

Linkage Disequilibrium analysis of a 210 kb region in the MHC class II region (Jeffreys et al., 2001) followed by sperm crossover analysis on a few males revealed five crossover hotspots of length 1-2 kb separated by long haplotype blocks containing tightly linked markers. This region contains another hotspot near the TAP2 gene identified earlier (Jeffreys et al., 2000) using the same technique. For this region, the locations of the sperm crossover hotspots and their intensities were in good agreement with the historical recombination rates estimated from coalescent analysis of the geno-

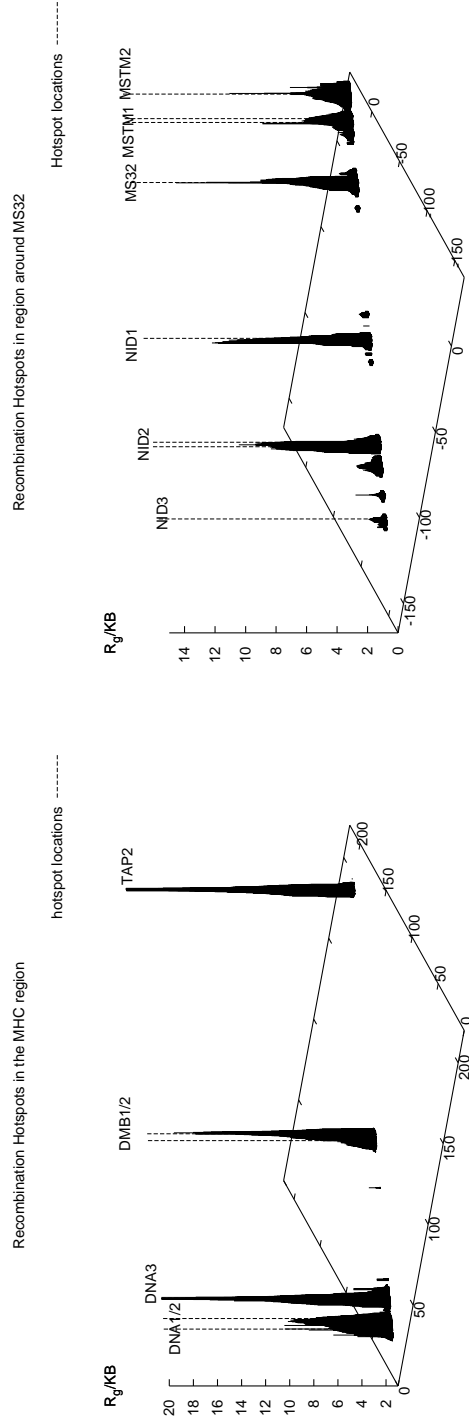


Figure 2.2: The density of detected recombination events in a 216KB segment of the class II region of MHC and a 206KB region on human chromosome 1 near the highly variable mini-satellite MS32. The vertical bars (labeled with the name of the corresponding hotspot) denote the approximate left and right edges of the hotspots detected using sperm crossover analysis. The plots were generated by considering all pairs of positions x and y in the region which were at least 0.5KB and at most 5KB apart. The z-axis is the number of detected recombination events (computed using the lower bound R_g) scaled by the length of the short segments.

type data (McVean et al., 2004). Note that the two methods measure different quantities; sperm analyses measures current recombination rates in males while population genetic methods estimate the sex-averaged recombination rates averaged over many generations. We also applied our lower bounds to the population data genotyped from the 50 UK individuals (Jeffreys et al., 2001). Since the data is unphased, we applied our lower bounds to the haplotypes estimated by the PHASE program (Stephens et al., 2001). In Figure 2.2 (top half of the figure), we plot recombination lower bounds for short segments (0.5KB to 5KB) from the MHC region. We use a simple statistic of plotting the recombination lower bound scaled by the length of the segment for which the bound was computed. (see (Myers and Griffiths, 2003)). The regions with high density of recombination events can easily be distinguished from the plot and correspond to putative recombination hotspots. We find that four hotspots: DNA2, DNA3, DMB2 and TAP2 are clearly identifiable. The hotspot DNA1 is difficult to distinguish from DNA2 since the centers of these hotspots are very close (3-4KB apart). Similarly the two hotspots DMB1 and DMB2 appear as a single cluster. It may be possible to further analyze the two regions to separate the hotspots. Note that the density of detected recombination events in a region is not directly interpretable in terms of the underlying recombination rate.

More recently, (Jeffreys et al., 2005) genotyped 200 SNP's for 80 UK males in a 206KB region of chromosome 1q4.23. LD analysis of this region revealed long haplotype blocks (upto 80kb) disrupted by five regions of low LD. Sperm crossover analysis of the regions with breakdown of LD identified seven new hotspots in addition to the MS32 hotspot. The genotype data was analyzed using three different coalescent based methods (Fearnhead et al., 2004; Li and Stephens, 2003; McVean et al., 2004). The LDHot method (McVean et al., 2004) found 4 hotspots with no false positives while the Hotspotter method (Li and Stephens, 2003) found 5 hotspots with 3 false positives. In comparison, the approximate likelihood method (Fearnhead et al., 2004) seemed the most accurate; it could detect 7 hotspots with a single false positive. The hotspots NID2 actually consists of two hotspots NID2a and NID2b and no coalescent based

method could separate these two hotspots. Similarly, the MSTM1 hotspot is a doublet of two closely spaced hotspots. As for the MHC region, we plotted recombination lower bounds for short segments for this region (see bottom of Figure 2.2). From the plot, one can visually identify five hotspots: NID2, NID1, MS32, MSTM1 and MSTM2. The hotspot NID3 is not detectable, a possible reason being the low recombination rate in this hotspot. From these results, it is clear that haplotype lower bounds can provide a first hand idea of the location and to some extent the intensities of most of the recombination hotspots detected using sperm crossover analysis. Although LD analysis can also identify many hotspots, the evidence for hotspots is much better from lower bounds than pairwise LD plots in some cases (see e.g. the β -globin hotspot (Fearnhead et al., 2004)).

In the remainder of this chapter, we present results for improving the complexity of the R_s lower bound and describe a new lower bound R_f that can detect more recombination events than either R_g or R_s for many datasets.

2.3 History Based Lower Bounds

(Myers and Griffiths, 2003) only give a procedural definition of the bound R_s , and their description is somewhat informal. The time complexity of their procedure (as described in Algorithm 3 in (Myers and Griffiths, 2003)) is $O(mn!)$, where n is the number of rows, and m the number of columns. We give a theoretical formulation of the bound R_s which allows us to develop an exponential time algorithm for computing it. We define a history for a set of n rows as simply an ordering of the rows. We start by redefining R_s in terms of appropriate cost of a row in a given history. Consider a history $H = r_1 \rightarrow r_2 \dots \rightarrow r_n$. The cost of row r_i in the history, denoted by $C_s(r_i)$, is 0 if after removing non-informative columns from r_1, r_2, \dots, r_i , the row r_i turns out to be identical to one of the rows r_1, \dots, r_{i-1} and 1 otherwise. Then we have

$$C_s(H) = \sum_i C_s(r_i) \text{ and } R_s(M) = \min_{\text{history } H} C_s(H)$$

We defer the discussion of why R_s , as we have defined it, is a lower bound to Theorem 2 (where we prove that R_I is a lower bound). Consider a bit vector \vec{r} of length n . Let $M_{\vec{r}}$ denote a submatrix of M which contains only rows i such that $r_i = 1$. Define a partial order on the vectors as follows: $\vec{v}_1 \leq \vec{v}_2$ if $v_2[i] = 1$ whenever $v_1[i] = 1$. Define the vector \vec{v}_{-i} as the \vec{v} with the i -th bit set to 0. Let $R_S[\vec{v}]$ denote the R_s bound for the corresponding sub-matrix.

Dynamic programming algorithm for computing $R_s(M)$:

1. **For all** row subsets \vec{r} : $R_S[\vec{r}] = 0$
2. **for all** subsets \vec{r} picked in an increasing order
3. **if** \exists a redundant row in \vec{r} : $R_S[\vec{r}] = R_S[\vec{r}_{-i}]$
4. **else** $R_S[\vec{r}] = \min_i \{1 + R_S[\vec{r}_{-i}]\}$ (* for all rows i s.t. $r_i = 1$ *)

The running time of the procedure above is $O(m2^n)$. Using a non-trivial reduction from the MAX-2SAT problem, we have shown that computing the bound R_s for a matrix is NP-hard (Bafna and Bansal, 2005).

The R_s bounds searches over possible histories of the set of haplotypes and one would expect the bound to be better than the diversity based bound R_h . However, in practice, R_s does not improve over R_h in most cases. Next, we describe a new lower bound R_I that improves over R_s .

2.3.1 Recombinant Intermediates and the bound R_I

We use an example to demonstrate how R_s can be improved. Consider the set of $n + 2$ haplotypes with n sites shown in Figure 2.3.1. For illustration $n = 7$.

Note that if the history was forced to start with the first two haplotypes, each of the following n rows could only be removed through a non-redundant row removal, and we would have a recombination bound of n . However, if we choose 1111111 to be the

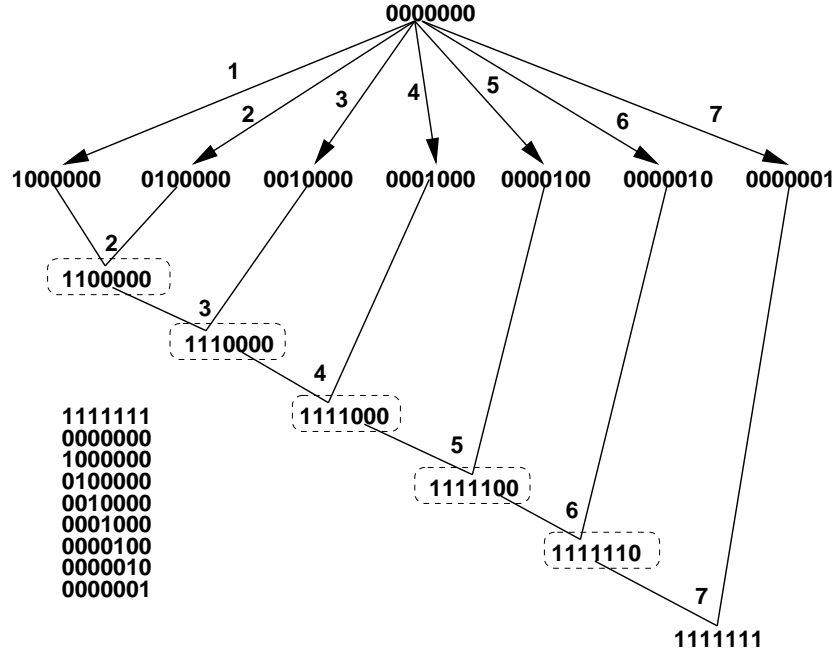


Figure 2.3: A set of 9 haplotypes for which R_s is 1 and a phylogenetic network for the set of haplotypes with 6 recombination events $R(I=6)$.

last haplotype in the history, then removing it makes every column non-informative. As R_s is the minimum over all histories, $R_s(M) = 1$. However, at least 6 recombinations are needed. Note that for this particular example, we can boost the R_s bound to the correct value by applying the dynamic programming algorithm (Myers and Griffiths, 2003) for combining local bounds. However, the example illustrates a problem with R_s , which is that in explaining a non-redundant row-removal, we only charge a *SINGLE* recombination event. Therefore, if 1111111 was indeed the last haplotype in the true history, then adding it would require 5 recombinants (the haplotypes in dashed boxes) NOT from the current set (as explained in Figure 2.3).

We use this idea to improve the R_S bound. Consider a history $H = r_1 \rightarrow r_2 \dots \rightarrow r_n$. Let $\mathcal{I}_j(H)$ denote the minimum number of recombination events in obtaining r_j , given any phylogenetic network for r_1, \dots, r_{j-1} . We allow the use of recombinant intermediates, and so $\mathcal{I}_j(H)$ can be greater than one. In general, the use of recombinant intermediates is tricky because the intermediates may help explain some

of the existing haplotypes by simple mutations. In order to prove a lower bound, we introduce the concept of a *direct recombination*. We define $C_d(r_i)$ for a haplotype r_i in a given history H as follows:

$$C_d(r_i) = \begin{cases} 0 & r_i \text{ is different from all } r_{j < i} \text{ in a non-informative column.} \\ 0 & r_i \text{ is identical to } r_{j < i} \text{ after removing non-informative columns} \\ 1 & \text{Otherwise} \end{cases} \quad (2.1)$$

We observe that the definition of $C_d(r_i)$ holds for a set of haplotypes $\{r_1, r_2, \dots, r_{i-1}, r_i\}$ and denote this generic definition as $C_d(r_i, \{r_1, r_2, \dots, r_{i-1}\})$. Note that $C_d(r_i) \leq C_s(r_i)$ for all i in a history. However, C_d can be used to give a new lower bound on the total number of recombinations.

Theorem 2: Let \mathcal{H} denote the set of all histories over the set of haplotypes M . Then

$$R_I = \min_{H \in \mathcal{H}} \max_j \left\{ \sum_{i < j} C_d(r_i) + \mathcal{I}_j(H) + \sum_{i \geq j} C_s(r_i) \right\}$$

is a lower bound on the number of recombinations.

Proof: Recall that m_M denotes the minimum number of recombinations in any history of M . We construct one history $H = r_1 \rightarrow r_2 \dots \rightarrow r_n$ in which which $\sum_{i < j} C_d(r_i) + \mathcal{I}_j(H) + \sum_{i > j} C_s(r_i)$ is a lower bound on m_M for all choices of j . This is sufficient because we minimize over all histories. Consider an phylogenetic network \mathcal{A} that explains m_M with a minimum number of recombinations. Each node v in the phylogenetic network corresponds to a haplotype r_v , which may or may not be in M . Haplotype $r \in M$ is a *direct witness* for a recombinant node v if $r = r_v$. It is an *indirect witness* if it can be derived from r_v solely by mutation events. A predecessor relationship $<_P$ is defined for some haplotypes $r_i, r_j \in M$. Specifically $r_i <_P r_j$ if r_i is a (direct or indirect) witness to a recombinant node on a path from the root to r_j . Note that $<_P$ is a partial order. Next, choose a history H (a total ordering) that is consistent with $<_P$. Note that $C_s(r_i) = 1$ if and only if r_i is the first witness to a recombination node in \mathcal{A} to appear in H (thereby proving that $R_s(M)$ is a lower bound). Likewise $C_d(r_i) = 1$ if

and only if r_i is the first direct witness to a recombination node in A to appear in H . As each recombination node contributes at most 1, $R_s = \sum_i C_s(r_i)$ is a valid lower bound on the number of recombinations. Consider an arbitrary r_j with $C_s(r_j) = 1$. Instead of charging 1 to the number of recombination events, we charge a value $\mathcal{I}_j(H)$ equal to the minimum number of recombinations needed to obtain r_j from r_1, r_2, \dots, r_{j-1} . Consider the sequence of intermediate recombination events that were used to obtain r_j . None of these nodes have a direct witness. Therefore the nodes in r_1, r_2, \dots, r_{j-1} that had a C_d value of 1 correspond to other recombination nodes.

Next, the haplotypes $r_{i>j}$ that follow r_i are charged $C_s(r_i)$. Whenever, $C_s(r_i) = 1$, it is because r_i is the first witness to a recombination node in A to appear in H . By construction, this recombination node is not on any path from root to r_j , and therefore wasn't charged when considering intermediates for r_j . Therefore, each recombination node is charged at most once and the bound holds. \clubsuit

The algorithm below describes how to compute R_I in time $O(n2^n I(m,n))$ time, where $I(m,n)$ is the time to compute $\mathcal{I}_j[\vec{r}]$ for any subset \vec{r} . $\mathcal{I}_i[\vec{r}_{-i}]$ denotes the minimum number of recombinant intermediates needed to compute haplotype r_i given the subset \vec{r} with r_i removed.

Dynamic programming algorithm for computing $R_I(M)$:

1. **For all** row subsets \vec{r} : $R_d[\vec{r}] = 0, R_I[\vec{r}] = 0$
2. **for all** subsets \vec{r} picked in an increasing order
3. **if** \exists a redundant row in \vec{r}
4. $R_d[\vec{r}] = \{R_d[\vec{r}_{-i}]\}; R_I[\vec{r}] = \{R_I[\vec{r}_{-i}]\}$
5. **else** for all rows i s.t. $r_i = 1$
6. $R_{d,i} = \min_i \{C_d(r_i, \vec{r}_{-i}) + R_d[\vec{r}_{-i}]\}; R_{I,i} = \min_i \{\max\{1 + R_I[\vec{r}_{-i}], R_d[\vec{r}_{-i}] + \mathcal{I}_i[\vec{r}_{-i}]\}\}$

7. $R_d[\vec{r}] = \min_i \{R_{d,i}\}; R_I[\vec{r}] = \min_i \{R_{I,i}\}$
8. return $R_I(M)$

It is easy to see that $R_I \geq R_s$. In order to compute R_I , we need to compute $\mathcal{I}_j(H)$ for all haplotypes j , and all histories H . To do this more efficiently, we define \mathcal{I}_j over subsets, instead of histories. We denote a subset of haplotypes by the bit-vector \vec{r} of size n where $\vec{r}_i = 1$ iff $r_i \in \vec{r}$ and define $\mathcal{I}_j[\vec{r}]$ as minimum number of recombination events needed to obtain r_j , over any history of the haplotypes in \vec{r} . Likewise, define $R_d(\vec{r})$ as the minimum number of direct recombinations in any history of the haplotype subset \vec{r} .

2.3.2 Computing Recombinant Intermediates

Our goal is to compute $\mathcal{I}_i[\vec{r}]$ efficiently. Haplotype i is assumed to arise later in history than in \vec{r} and is therefore a mosaic of sub-intervals of the haplotypes in \vec{r} . The mosaic can be expressed by a sequence of pairs $M = (h_1, j_1), (h_2, j_2) \dots, (h_k, j_k)$ interpreted as follows: In h_i , columns $1, \dots, j_1$ came from haplotype h_1 , columns $j_1 + 1, \dots, j_2 + 1$ from h_2 , and so on. If M were the true mosaic, then h_i would need $k - 1$ recombinant intermediates. Thus, we need to minimize this.

First, we can ignore all columns that are identical for all haplotypes in \vec{r} . If h_i has a different value in any of these columns, it can be explained by a mutation. If it has the identical value, the column can be explained using any haplotype and will not contribute to recombination. Ignoring these columns, the following is true: if columns j_1, \dots, j_2 of h_i arise from haplotype h , then the values of h and h_i must be identical in columns j_1 through j_2 . If any column c was different ($h_i[c] \neq h[c]$), to explain it by a mutation would violate the infinite-sites assumption. This observation allows us to solve the problem of computing $\mathcal{I}_i[\vec{r}]$ efficiently.

For column c , $1 \leq c \leq m$ and haplotype h , let $I[c, h]$ denote the minimum number of recombinations needed to explain the first c columns of haplotype h_i such that the

c -th column arose from haplotype h . This is sufficient because $\mathcal{I}_i[\vec{r}] = \min_h \{I[m, h]\}$. $I[c, h]$ can be computed using the following recurrence:

$$I[c, h] = \begin{cases} 0 & c = 0 \\ \infty & h_i[c] \neq h[c] \\ \min \{I[c-1, h], \min_{h' \neq h} \{1 + I[c-1, h']\}\} & o/w \end{cases}$$

2.3.3 Results for R_I bound

Besides the simulated example (in Figure 2.3), real datasets are known where R_s and R_h are sub-optimal. As an example, the R_h and R_s bounds for Kreitman's data (Kreitman, 1983) from the ADH locus of *Drosophila Melanogaster* are both 6. Song and Hein (Song and Hein, 2004) showed that their set theoretic lower bound gave a bound of 7 and proved this to be optimal by actually constructing an phylogenetic network which requires 7 recombination events. Our new lower bound R_I also returns the optimal bound of 7. However, the set theoretic-bound (Song and Hein, 2004) does not have an explicit algorithmic description. On the other hand, the R_I bound can be computed for large datasets (100×500 matrix can be analyzed in few hours on a standard PC) and gives improved bounds for a number of real datasets (see table 2.3 for a partial list).

2.4 Discussion and Future Work

We have presented new computational methods for computing lower bounds on the minimum number of recombination events from a sample of haplotypes. We have shown that one of these lower bounds is very fast to compute and more sensitive to changes in recombination rate than previous bounds. Plots of this lower bound for two regions from the human genome for which recombination hotspots have been identified experimentally, provide a strong signal for most of the detected hotspots.

Table 2.3: Comparison of the number of detected recombination events using R_s and R_I for the phased haplotype datasets for various genes obtained from the SeattleSNP project (SeattleSNPs, <http://pga.gs.washington.edu>, March 2004).

Dataset	Size	R_s	R_I
CSF3	15 x 17	3	4
MMP3	21 x 41	6	9
ABO	68 x 197	70	73
DCN	31 x 117	16	18
HMOX1	34 x 53	14	16
F2RL3	28 x 29	10	11
EPHB6	31 x 62	23	25

There is an inherent stochasticity in the number of historical recombination events for a region of fixed length given a fixed recombination rate. This stochasticity is independent of the number of segregating sites in the region. On the other hand, any method for detecting historical recombination events is highly dependent on the number of mutations (segregating sites). In the worst case of no mutations, no method can detect any recombinations. Furthermore, the power to detect historical recombination events depends greatly on ancient mutations, i.e. mutations which happened before the recombination events. Note that any mutation that happened after a particular recombination event is non-informative for detecting that particular recombination event. It is interesting to note that (Jeffreys et al., 2001) choose markers with high rare allele frequency (> 0.15) under the assumption that these are likely to be ancient and hence provide greater evidence for breakage of haplotypes by recombination events. Therefore, one needs to incorporate this bias for high frequency mutations in methods which use summary statistics for estimating recombination rates using coalescent simulations (Wall, 2000). This will possibly provide greater power to detect hotspots when these methods

are applied to real datasets. We are currently pursuing this line of research.

Previous papers (Myers and Griffiths, 2003) have suggested that the minimum number of recombination events will miss most historical recombination events and should not be used as an indicator of the true number of historical recombination events or directly used to estimate the recombination rate. One should use full likelihood methods (Fearnhead and Donnelly, 2001) or approximate likelihood methods (Fearnhead et al., 2004; Li and Stephens, 2003; Hudson, 2001) for estimation the underlying recombination rate. However, the computational burden imposed by these methods can be sometimes prohibitive for large scale datasets. Results of (Wall, 2000; Hudson, 2001) suggest that estimates of the recombination rate obtained in a maximum-likelihood framework using easy to compute summary statistics of the data, are comparable to estimates using pairwise likelihoods (Hudson, 2001; McVean et al., 2002). Therefore, it remains to be seen whether using new summary statistics for recombination (such as lower bounds described in this chapter) and correcting for the bias in the number of mutations and mutations with low rare allele frequency, one can obtain good estimates of the recombination rates and detect recombination hotspots reliably.

From the computational standpoint, there are several open questions. How close are these recombination lower bounds to the minimum number of recombination events for a sample ? We believe that the difference between the lower bounds and the true minimum could increase as the recombination rate increases, although it is difficult to verify this and may not be true. The recombination lower bounds described here and almost all previous lower bounds are applicable to haplotype data. Unfortunately it is experimentally difficult and expensive to obtain phase information for genotypes. We obtain nice results for recombination hotspots using computationally phased haplotype data. However, computing recombination lower bounds from unphased data remains an interesting computational problem (Wiuf, 2004).

2.5 Acknowledgement

Chapter 2, in full, was published in the Journal of Computational Biology, Vol 13(2), pp 501-21, 2006, V. Bafna and V. Bansal, “Inference about Recombination from Haplotype Data: Lower Bounds and Recombination Hotspots”. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Counting Recombination Events: Conflict Graph and Lower Bounds

In the previous chapter, we investigated the computational complexity of two lower bounds on the minimum number of recombination events. We also described efficient algorithms for computing these lower bounds and a new lower bound that is better than all previous lower bounds. All of these recombination bounds are explicitly or implicitly based on haplotype diversity. The lower bound of Hudson and Kaplan (1985) is based on estimating the maximum possible number of conflicting pairs of sites. The conflict graph for a set of sequences represents the pairs of sites that do not satisfy the four-gamete test. The conflict graph was used by D.Gusfield et al. (2003) to obtain a polynomial time algorithm for the galled tree problem, which is a special case of the Ancestral Recombination Graph (ARG) reconstruction problem. In this chapter, we show that the number of non-trivial connected components in the conflict graph for a given set of sequences is a lower bound on the minimum number of recombination events required to explain the set of sequences under the infinite sites model of evolution. We show that in many cases, this lower bound, R_c is a better bound than the haplotype lower bound R_h . Our results also offer some insight into the structural properties of this graph and are of interest for the general ARG reconstruction problem.

In particular, we demonstrate the following results:

1. The number of non-trivial connected components R_c in the conflict graph of a set of haplotypes is a lower bound on the number of recombinations required to explain the history of the sample under the infinite-sites model. This bound can be computed in $O(nm^2)$ time for a matrix of size $n \times m$.
2. The number of recombinations required does not increase if sites are deleted. Based on this idea, we introduce the *Max-NTCC* problem of deleting sites so that the number of non-trivial connected components is maximized (to give improved lower bounds). We show that this problem is NP-complete using a reduction from the Maximum Independent Set problem.
3. We show that for any set of haplotypes, $R_s \geq R_c$. Although R_c is generally a weaker lower bound than R_h , there are many instances where it can be greater than R_h . We show that for any matrix M , $R_h \geq \frac{2}{3}R_c - \frac{1}{3}$ and also provide infinitely many examples for which $R_h = \frac{2}{3}R_c$. Additionally, we show how R_c can be combined with the other bounds to reduce the computation time in practice, and present a real dataset where it offers improvements.

3.1 Definitions

We will use the same notation and definitions as used in the previous chapter. We provide some definitions of conflicts and the conflict graph introduced by D.Gusfield et al. (2003).

Definition 1: A pair of columns (i, j) in M are said to *conflict* if there is set of 4 rows with the pairs $\{00, 01, 10, 11\}$ in these two columns. If the ancestral type at each site is known and assumed to be 0, the presence of three rows with the values $\{01, 10, 11\}$ in these two columns implies a conflict, since we can infer the existence of the ancestral type 00. A pair of columns (i, j) is said to be *compatible* if i and j do not conflict.

Definition 2: [from D.Gusfield et al. (2003)] The conflict graph $G_C(M) = (V, E)$ for a given set M of n sequences is a graph with vertex set $V = \{i \mid i \text{ is a column of } M\}$ and $E = \{(i, j) \mid \text{columns } i \text{ and } j \text{ conflict}\}$. Note that matrix M defines an ordering for the vertices of $G_C(M)$. We define two edges (a, b) and (c, d) in $G_C(M)$ to be *non-interleaving* if $\max\{a, b\} < \min\{c, d\}$ or $\max\{c, d\} < \min\{a, b\}$.

Definition 3: We define $R_c(M)$ to be the number of non-trivial connected components (components of size more than one) in the conflict graph of a set of sequences M .

Definition 4: Let M be an $n \times m$ matrix, and $S = \{1, 2, \dots, m\}$ denote the set of sites in M . For a subset $S' \subseteq S$, let $M(S')$ denote the submatrix obtained by restricting columns to be in S' , and removing all redundant rows. For a site s , denote $M(S - \{s\})$ by M_{-s} .

3.2 Connected Components in the Conflict Graph

We begin by showing that removing sites from the matrix corresponding to the given set of sequences does not increase the required number of recombination events.

Lemma 3: For any subset $S' \subseteq \{1, 2, \dots, m\}$ of the columns of a matrix M , $m_{M(S')} \leq m_M$, and therefore any lower bound on the number of recombinations for the matrix $M(S')$ is also a lower bound on m_M .

Proof: Consider an ancestral recombination graph $G(M)$ explaining M . For any arbitrary site $s \in S$, we show how to transform $G(M)$ into a ancestral recombination graph explaining M_{-s} without increasing the number of recombination cycles in $G(M)$. Consider the edge $e = (u, v)$ labeled with s in $G(M)$. If the edge e is labeled with other sites as well, we simply delete the label s and keep the ancestral recombination graph unchanged. Therefore, the interesting case is when s is the only label on the edge e . If v is a leaf node, we simply remove e and the node v . Clearly, this can only happen if the

sequence labeling the leaf node was the only sequence in M with a mutation on c , and hence is not present in M_{-s} .

In the case where v is an internal node in $G(M)$, we collapse the edge e and make u to be the starting vertex of all outgoing edges from v . It is easy to see that collapsing the edge e does not induce a cycle in the graph $G(M)$. Hence, the modified graph is an ARG for the matrix M_{-s} . Using an inductive argument, it is easy to see that

$$m_{M(S')} \leq m_M \quad \forall S' \subseteq S$$



Corollary 4: Let S' denote a subset of the sites S in M . Then $\max_{S' \subseteq S} m_{M(S')}$ is a lower bound on m_M .

Lemma 5: For a matrix M and a set of sites $S' \subseteq S$, $R_s(M(S')) \leq R_s(M) \leq m_M$.

Proof: Observe that any sequence of non-informative column deletions, row deletions and non-redundant row removal (from the definition of bound R_s) operations that reduces the matrix M to an empty matrix, also reduces the matrix $M(S')$ to an empty matrix. Hence, $R_s(M(S'))$ which is the number of row removal operations in a sequence which uses the minimum number of row removal operations, is at most $R_s(M)$.



The main idea behind the proof of the connected components lower bound is based on the computation of the lower bound R_s . In the R_s computation, we delete rows and columns, but we only charge a recombination event when we are deleting a non-redundant row. We will show that a row that is non-redundant when restricted to sites in a connected component MUST be redundant when restricted to sites of any other connected component. In order to show this, we must prove a structural property of two connected components described in the *2 edge theorem (Theorem 8)*. The proof of this theorem depends on two technical lemmas which we prove next.

Lemma 6: Let $S = \{i, j, k, l\}$ be four columns in a matrix M such that the pairs (i, j) and (k, l) conflict and (i, k) , (i, l) , (j, k) and (j, l) are compatible. Then there is at most one pair $a \in \{00, 01, 10, 11\}$ such that $[ab_1]$ and $[ab_2]$ are distinct rows in $M(S)$, where $b_1, b_2 \in \{00, 01, 10, 11\}$.

Proof: The proof is by contradiction. Suppose there are 4 distinct rows $[a_1b_1]$, $[a_1b_2]$, $[a_2b_3]$, $[a_2b_4]$ in $M(S)$. Without loss of generality, we can assume that $a_1 = b_1 = 00$ (we can relabel the columns without changing the conflict graph). We now consider two cases where $a_2 = 01$ and $a_2 = 11$. Since the ordering of the columns is not important, the case where $a_2 = 10$ is the same as $a_2 = 01$.

Case $a_2 = 01$: Since $b_3 \neq b_4$, they differ in at least one of the sites $\{k, l\}$. We can assume that they differ in the column k (as the order of the columns is not relevant). Also, as (i, j) conflict, the rows $a_3 = 10$, and $a_4 = 11$ exist. The remaining entries in the matrix are constrained to have particular values (see Figure 3.1). For the last matrix in Figure 3.1, we observe that the pair of columns (j, l) contains the four distinct pairs $00, 01, 1z$, and $1\bar{z}$. This is a contradiction to the fact that the pair of columns does not conflict.

Case $a_2 = 11$: The argument for this case proceeds along the same lines as the previous one. Since the only conflicts possible are between the sites (i, j) and (k, l) the submatrix is constrained to contain the following distinct set of rows:

i	j	k	l
0	0	0	0
0	0	0	1
1	1	1	z
1	1	0	z
1	0	0	z
0	1	0	z
x	y	1	\bar{z}

If $y = 0$, then (j, k) conflict. If $y = 1$, then (j, l) conflict, a contradiction! ♣

Lemma 7: Let $S = \{i, j, k, l\}$ be four columns in a matrix M such that (i, j) and (k, l) conflict and (i, k) , (i, l) , (j, k) and (j, l) are compatible. There the submatrix $M(S)$

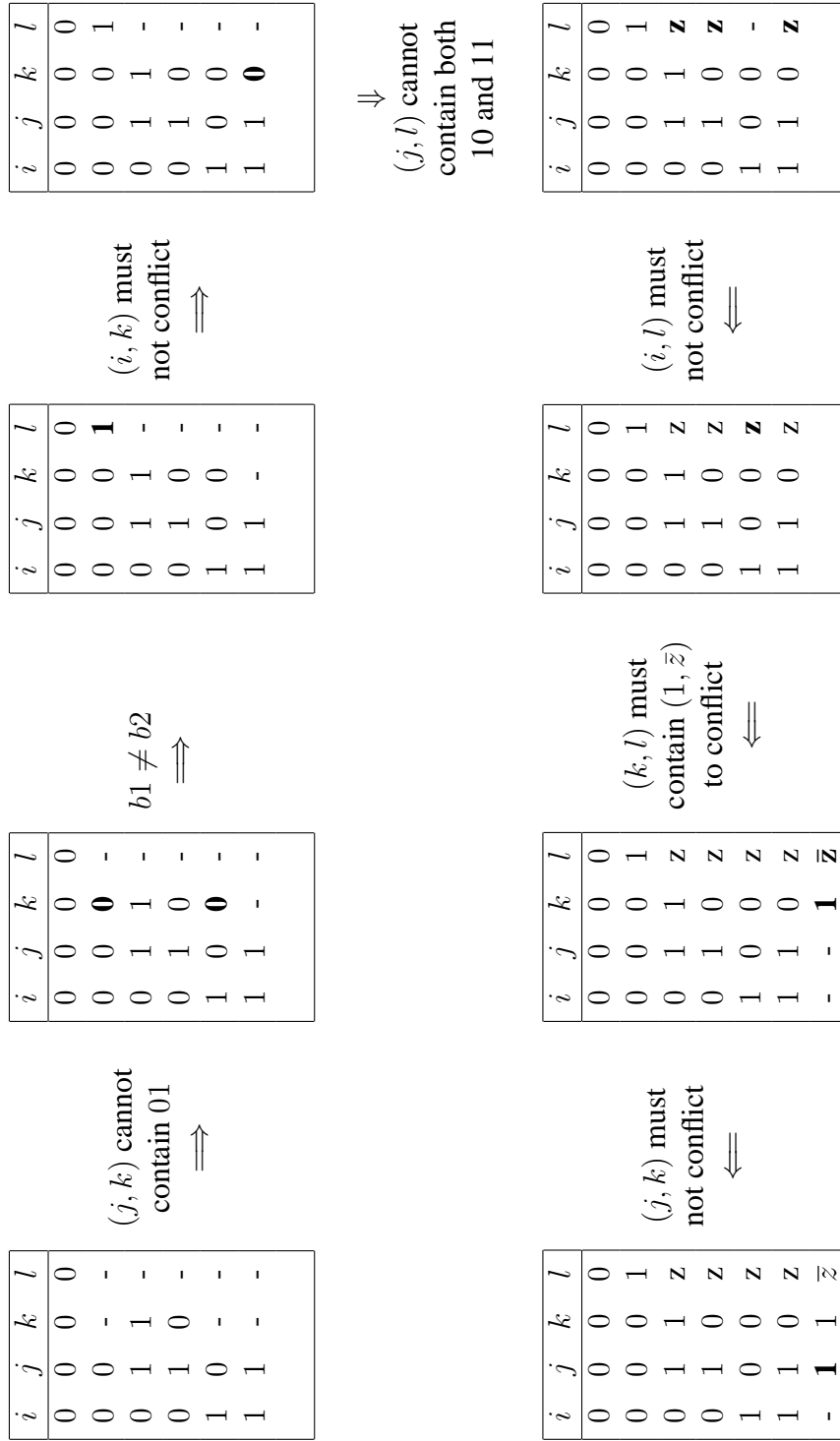


Figure 3.1: A step by step illustration of how the conditions of Lemma 3 constrain the matrix to have particular values, leading to a contradiction.

does not have three distinct rows of the form $[a_1b_1], [a_2b_2], [a_3b_3]$ where $a_1 \neq a_2 \neq a_3$ and $b_1 \neq b_2 \neq b_3$.

Proof: Suppose to the contrary we have 3 distinct rows $[a_1b_1], [a_2b_2], [a_3b_3]$ in $M(S)$. Without loss of generality we can assume that $a_1 = b_1 = 00$ and $a_2 = 01$ (we can relabel the columns without changing the conflict graph and the ordering of the rows and columns is not important). As before, we proceed using a case-by-case analysis.

Case: $a_3 = 11$ and $b_2 = 01$:

<table style="border-collapse: collapse; width: 100%;"> <thead> <tr><th>i</th><th>j</th><th>k</th><th>l</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>-</td></tr> <tr><td>1</td><td>0</td><td>-</td><td>-</td></tr> </tbody> </table>	i	j	k	l	0	0	0	0	0	1	0	1	1	1	1	-	1	0	-	-	(j, k) can't contain 01 & (k, l) conflict \implies	<table style="border-collapse: collapse; width: 100%;"> <thead> <tr><th>i</th><th>j</th><th>k</th><th>l</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>z</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>-</td></tr> <tr><td>-</td><td>1</td><td>1</td><td>\bar{z}</td></tr> </tbody> </table>	i	j	k	l	0	0	0	0	0	1	0	1	1	1	1	z	1	0	0	-	-	1	1	\bar{z}	(i, k) can't conflict \implies	<table style="border-collapse: collapse; width: 100%;"> <thead> <tr><th>i</th><th>j</th><th>k</th><th>l</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>z</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>-</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>\bar{z}</td></tr> </tbody> </table>	i	j	k	l	0	0	0	0	0	1	0	1	1	1	1	z	1	0	0	-	1	1	1	\bar{z}
i	j	k	l																																																																					
0	0	0	0																																																																					
0	1	0	1																																																																					
1	1	1	-																																																																					
1	0	-	-																																																																					
i	j	k	l																																																																					
0	0	0	0																																																																					
0	1	0	1																																																																					
1	1	1	z																																																																					
1	0	0	-																																																																					
-	1	1	\bar{z}																																																																					
i	j	k	l																																																																					
0	0	0	0																																																																					
0	1	0	1																																																																					
1	1	1	z																																																																					
1	0	0	-																																																																					
1	1	1	\bar{z}																																																																					

But now (i, l) contains 00, 01, $1z$, and $1, \bar{z}$, a contradiction!

Case: $a_3 = 11$ and $b_2 = 11$:

<table style="border-collapse: collapse; width: 100%;"> <thead> <tr><th>i</th><th>j</th><th>k</th><th>l</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>-</td><td>-</td></tr> </tbody> </table>	i	j	k	l	0	0	0	0	0	1	1	1	1	1	0	1	1	0	-	-	(j, k) cannot contain 01 & (k, l) conflict \implies	<table style="border-collapse: collapse; width: 100%;"> <thead> <tr><th>i</th><th>j</th><th>k</th><th>l</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>-</td></tr> <tr><td>-</td><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	i	j	k	l	0	0	0	0	0	1	1	1	1	1	0	1	1	0	0	-	-	1	1	0	(i, l) must not conflict \implies	<table style="border-collapse: collapse; width: 100%;"> <thead> <tr><th>i</th><th>j</th><th>k</th><th>l</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>-</td><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	i	j	k	l	0	0	0	0	0	1	1	1	1	1	0	1	1	0	0	1	-	1	1	0
i	j	k	l																																																																					
0	0	0	0																																																																					
0	1	1	1																																																																					
1	1	0	1																																																																					
1	0	-	-																																																																					
i	j	k	l																																																																					
0	0	0	0																																																																					
0	1	1	1																																																																					
1	1	0	1																																																																					
1	0	0	-																																																																					
-	1	1	0																																																																					
i	j	k	l																																																																					
0	0	0	0																																																																					
0	1	1	1																																																																					
1	1	0	1																																																																					
1	0	0	1																																																																					
-	1	1	0																																																																					

Now, the pair of columns (j, l) conflict, which is a contradiction.

Note that, since the ordering of the columns does not matter, the cases where $b_2 = 10$ is identical to the case where $b_2 = 01$.

Case: $a_3 = 10$ and $b_2 = 01$:

<table style="border-collapse: collapse; width: 100%;"> <tr><th style="padding: 2px 5px;">i</th><th style="padding: 2px 5px;">j</th><th style="padding: 2px 5px;">k</th><th style="padding: 2px 5px;">l</th></tr> <tr><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td></tr> <tr><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td></tr> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">-</td></tr> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">-</td><td style="padding: 2px 5px;">-</td></tr> </table>	i	j	k	l	0	0	0	0	0	1	0	1	1	0	1	-	1	1	-	-	(j, k) cannot contain 01 & (k, l) conflict \implies	<table style="border-collapse: collapse; width: 100%;"> <tr><th style="padding: 2px 5px;">i</th><th style="padding: 2px 5px;">j</th><th style="padding: 2px 5px;">k</th><th style="padding: 2px 5px;">l</th></tr> <tr><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td></tr> <tr><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td></tr> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">z</td></tr> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">-</td></tr> <tr><td style="padding: 2px 5px;">-</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">\bar{z}</td></tr> </table>	i	j	k	l	0	0	0	0	0	1	0	1	1	0	1	z	1	1	0	-	-	1	1	\bar{z}	(i, k) cannot conflict \implies	<table style="border-collapse: collapse; width: 100%;"> <tr><th style="padding: 2px 5px;">i</th><th style="padding: 2px 5px;">j</th><th style="padding: 2px 5px;">k</th><th style="padding: 2px 5px;">l</th></tr> <tr><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td></tr> <tr><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td></tr> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">z</td></tr> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">-</td></tr> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">\bar{z}</td></tr> </table>	i	j	k	l	0	0	0	0	0	1	0	1	1	0	1	z	1	1	0	-	1	1	1	\bar{z}
i	j	k	l																																																																					
0	0	0	0																																																																					
0	1	0	1																																																																					
1	0	1	-																																																																					
1	1	-	-																																																																					
i	j	k	l																																																																					
0	0	0	0																																																																					
0	1	0	1																																																																					
1	0	1	z																																																																					
1	1	0	-																																																																					
-	1	1	\bar{z}																																																																					
i	j	k	l																																																																					
0	0	0	0																																																																					
0	1	0	1																																																																					
1	0	1	z																																																																					
1	1	0	-																																																																					
1	1	1	\bar{z}																																																																					

But now (i, l) contains 00, 01, 1 z , and 1, \bar{z} , hence a conflict.

Case: $a_3 = 10$ and $b_2 = 11$:

<table style="border-collapse: collapse; width: 100%;"> <tr><th style="padding: 2px 5px;">i</th><th style="padding: 2px 5px;">j</th><th style="padding: 2px 5px;">k</th><th style="padding: 2px 5px;">l</th></tr> <tr><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td></tr> <tr><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td></tr> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td></tr> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">-</td><td style="padding: 2px 5px;">-</td></tr> </table>	i	j	k	l	0	0	0	0	0	1	1	1	1	0	0	1	1	1	-	-	(i, k) cannot contain 11 & (k, l) conflict \implies	<table style="border-collapse: collapse; width: 100%;"> <tr><th style="padding: 2px 5px;">i</th><th style="padding: 2px 5px;">j</th><th style="padding: 2px 5px;">k</th><th style="padding: 2px 5px;">l</th></tr> <tr><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td></tr> <tr><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td></tr> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td></tr> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">-</td></tr> <tr><td style="padding: 2px 5px;">-</td><td style="padding: 2px 5px;">-</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td></tr> </table>	i	j	k	l	0	0	0	0	0	1	1	1	1	0	0	1	1	1	0	-	-	-	1	0	(j, k) must not conflict \implies	<table style="border-collapse: collapse; width: 100%;"> <tr><th style="padding: 2px 5px;">i</th><th style="padding: 2px 5px;">j</th><th style="padding: 2px 5px;">k</th><th style="padding: 2px 5px;">l</th></tr> <tr><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td></tr> <tr><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td></tr> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td></tr> <tr><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">-</td></tr> <tr><td style="padding: 2px 5px;">-</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td></tr> </table>	i	j	k	l	0	0	0	0	0	1	1	1	1	0	0	1	1	1	0	-	-	1	1	0
i	j	k	l																																																																					
0	0	0	0																																																																					
0	1	1	1																																																																					
1	0	0	1																																																																					
1	1	-	-																																																																					
i	j	k	l																																																																					
0	0	0	0																																																																					
0	1	1	1																																																																					
1	0	0	1																																																																					
1	1	0	-																																																																					
-	-	1	0																																																																					
i	j	k	l																																																																					
0	0	0	0																																																																					
0	1	1	1																																																																					
1	0	0	1																																																																					
1	1	0	-																																																																					
-	1	1	0																																																																					

Now, the pair of columns (j, l) conflict, which is a contradiction. This completes the proof of the lemma. ♣

Theorem 8: [2-edge theorem] Let $S = \{i, j, k, l\}$ be four columns in a matrix M such that (i, j) and (k, l) conflict and (i, k) , (i, l) , (j, k) and (j, l) are compatible. Then there exists pairs a_{ij} and b_{kl} , such that every row in the submatrix $M(S)$ is of the form $[a_{ij}b]$ or $[ab_{kl}]$ where $a_{ij}, b_{kl}, a, b \in \{00, 01, 10, 11\}$.

Proof: Consider 3 distinct rows in the submatrix $M(S)$ of the form $[a_1b_1], [a_2b_2], [a_3b_3]$ where $a_1 \neq a_2 \neq a_3$. From Lemma 7 it follows that it cannot be the case that $b_1 \neq b_2 \neq b_3$.

First, we consider the special case where $b_1 = b_2 = b_3$. Then consider the 3 rows $[a_4b_4], [a_5b_5], [a_6b_6]$ where $b_4 \neq b_5 \neq b_6 \neq b_1$. Since the columns (k, l) conflict, three such rows exist. Now if $a_4 = a_5 = a_6$, then there cannot be a row of the form $[ab]$ where $a \in \{a_1, a_2, a_3\}$ and $b \neq b_1$, since then we would have 4 distinct rows $[ab_1], [ab], [a_4b_4], [a_4b_5]$ which violate lemma 6. Hence, the only other row we can have

is $[a_4b_1]$ and therefore the lemma holds with $a_{ij} = a_4$ and $b_{kl} = b_1$. If we have $a_4 = a_5 \neq a_6$, then we cannot have another row of the form $[ab]$ where $a \neq a_4$, since then we would again have 4 distinct rows violating the constraint of lemma 6. Hence, the lemma is satisfied with $a_{ij} = a_4$ and $b_{kl} = b_1$.

In the case where $b_1 = b_2 \neq b_3$, lemma 6 enforces that there is exactly one row of the form $[a_4b_4]$ where b_4 is different from both b_1 and b_3 and $a_4 \in \{00, 01, 10, 11\}$. Similarly, there is one and only one row of the form $[a_5b_5]$ where b_5 is different from either of b_1, b_3 or b_4 and $a_5 \in \{00, 01, 10, 11\}$. If $a_4 \neq a_3$ and $a_4 \neq a_2$, then the three rows $[a_2b_1], [a_3b_3], [a_4b_4]$ violate lemma 7. Applying lemma 7 to the three rows $[a_1b_1], [a_3b_2], [a_4b_4]$, we get that either $a_4 = a_1$ or $a_4 = a_3$. But both the previous constraints can be satisfied if and only if $a_4 = a_3$, since $a_1 \neq a_2$. Using similar arguments, it follows that $a_5 = a_3$. Hence, the lemma is true with $a_{ij} = a_3$ and $b_{kl} = b_1$.



Definition 5: Let $S = \{i, j, k, l\}$ be a subset of columns of a matrix M such that the pairs (i, j) and (k, l) conflict and $(i, k), (i, l), (j, k), (j, l)$ are compatible. Then we denote by a_{ij} and b_{kl} the pairs, such that every row in the submatrix $M(S)$ is of the form $[a_{ij}b]$ or $[ab_{kl}]$.

The next lemma explains how a non-redundant row removal (a row removal is said to be non-redundant when a column deletion or a row deletion cannot be done) can only destroy one connected component. The connected component theorem will follow from a simple application of this lemma and Lemma 3.

Lemma 9: Consider a matrix M such that each connected component in the conflict graph $G_C(M)$ has size 2, i.e. it consists of two sites which are in conflict. Then $R_s(M) \geq R_c(M)$.

Proof: We show that every possible sequence of non-redundant row removal events which reduces a matrix M (whose conflict graph has the structure described above) to the empty matrix, requires at least $R_c(M)$ non-redundant row removal operations.

Since, the bound R_s is the minimum number of row removal operations performed for some sequence of non-redundant row removal events, it follows that $R_s(M) \geq R_c(M)$. We claim that any column or row deletion cannot remove an edge in the conflict graph (or equivalently destroy a connected component). A site is deleted only when it is non-informative, i.e. there is either only a single haplotype with a 1 at that site or a single haplotype with a 0 at that site. Clearly, such a site cannot be involved in a conflict with another site, since one needs at least 2 ones and 2 zeroes at a site for it to be involved in a conflict. Similarly, a row is deleted when there are two identical haplotypes. Clearly, a removal of one of them cannot remove a conflict between two sites.

Now, consider a non-redundant row removal operation which destroys a conflict between two sites (i, j) , i.e it removes one of pairs $\{00, 10, 01, 11\}$ from the two columns. Denote the pairs removed by (ab) . Consider a pair of conflicting sites (k, l) and the submatrix $M(S)$ restricted to the 4 sites $S = \{i, j, k, l\}$. Clearly $ab \neq a_{ij}$ (where a_{ij} is as defined above), since a_{ij} is present in more than one row of $M(S)$. From the 2-edge lemma it follows that the pair in the columns (k, l) in the row that was removed is also present in other rows in $M(S)$. Hence, the removal of the row containing (ab) in the columns (i, j) cannot destroy a conflict between the sites (k, l) . Hence, a non-redundant row removal operation can destroy at most one connected component (or conflict) in the conflict graph. Therefore, by induction, any sequence of column deletions, row deletions, or non-redundant row removal events requires at least $R_c(M)$ non-redundant row removals to reduce the matrix to the empty matrix. ♣

Theorem 10: For every matrix M , $R_c(M) \leq R_s(M) \leq m_M$.

Proof: For every non-trivial connected component in $R_c(M)$, we remove sites such that only two conflicting sites remain in each connected component. Clearly, we can do this for every connected component with 2 or more sites. Hence, after removal of a subset of sites, we have a reduced matrix $M(S')$ where S' is the set of remaining sites. From lemma 5, $m_M \geq R_s(M(S'))$. Also from lemma 9, it follows that $R_s(M(S')) \geq R_c(M)$, which proves the required result.

Note that the 2-edge theorem (Theorem 8) imposes a strong structure on the underlying matrix. We can extend this theorem to the general case where we have a connected component instead of an edge. The theorem below, shows that the rows conferring haplotype diversity to a connected component are disjoint for each connected component.



Theorem 11: Let A, B be disjoint subsets of columns representing distinct connected components in the conflict graph. Let a_1, \dots, a_k and b_1, \dots, b_l be the distinct rows (haplotypes) in $M(A)$ and $M(B)$ respectively. There exist haplotypes a_i and b_j , such that all distinct rows of the matrix $M[A \cup B]$ are of the type $[a_i b]$ for some $b \in \{b_1, \dots, b_l\}$, or $[a b_j]$ for some $a \in \{a_1, \dots, a_k\}$.

Proof: We prove by induction on the total number of columns in M . As the two components are non-trivial, M has at least 4 columns containing an edge in each component.

Base case (4 columns): Each component has 4 distinct haplotypes 00, 01, 10, and 11. The base case follows directly from Theorem 8.

Induction step ($k + 1$ columns): Assume that the hypothesis is true for all matrices containing two non-trivial components with a total of k columns. Let A be the component with the larger number of columns. Remove a column i from A to get A' , such that the columns in A' still form a single connected component¹.

By the induction hypothesis, there exist haplotypes a'_i and b_j , such that all distinct rows of $M[A' \cup B]$ are of the type $[a'_i b]$ for some $b \in \{b_1, \dots, b_l\}$, or $[a b_j]$ for some $a \in \{a_1, \dots, a_k\}$. Thus, the distinct rows of $M(A' \cup B)$ are:

¹ such a column always exists, since one can remove a vertex from a connected graph such that the remaining vertices still form a connected graph

A'	B
a'_i	b_1
a'_i	b_2
\vdots	
a'_i	b_l
a'_1	b_j
a'_2	b_j
\vdots	
a'_k	b_j

Now add the i -th column back to get $M(A \cup B)$. Consider all the rows containing a'_i . We claim that all rows containing a'_i must have the same value z in the i^{th} column. If this is true, the rows of $M(A \cup B)$ are

i	A'	B
z	a'_i	b_1
z	a'_i	b_2
	\vdots	
z	a'_i	b_l
	a'_1	b_j
	a'_2	b_j
	\vdots	
	a'_k	b_j

Let $a_i = [z a'_i]$. Then each row is of the form $[a_i b]$, or $[a b_j]$, and we are done.


Next, consider the case when the rows containing a'_i have instances of z and \bar{z} in the i -th column. Without loss of generality, rename the haplotypes of B so that the rows of $M(A \cup B)$ contain

i	A'	B
z	a'_i	b_1
\bar{z}	a'_i	b_2
	\vdots	
	a'_i	b_l
	a'_1	b_j
	a'_2	b_j
	\vdots	
	a'_k	b_j

Next, consider an arbitrary column $j \in A'$ such that i, j conflict, and denote the value of row a'_i in column j as x . Consider the connected component $X = \{i, j\}$, and B . As

(i, j) conflict, all four rows $zx, \bar{z}x, z\bar{x}, \bar{z}\bar{x}$ must appear. On the other hand, as a'_i only contains x , all rows containing \bar{x} must line up against b_j . The columns of $M(X \cup B)$ contain

X	B
zx	b_1
$\bar{z}x$	b_2
\vdots	
	b_l
$z\bar{x}$	b_j
$\bar{z}\bar{x}$	b_j
\vdots	
	b_j

It is easy to verify that the components X and B violate the inductive hypothesis even though they $X \cup B$ has fewer than k columns, a contradiction! 

Definition 6: For a pair of non-trivial connected components (A, B) , denote the common haplotype of A with respect to B as $h(A, B)$.

3.2.1 Extensions to the R_c lower bound

In this subsection, we show how the connected component lower bound can be extended to obtain improved bounds. We show that we can apply the lower bound R_s independently to each connected component of the conflict graph of a matrix M to obtain a recombination lower bound for M .

Lemma 12:

$$\sum_{C \in \mathcal{CC}} R_h(M(C)) \leq \sum_{C \in \mathcal{CC}} R_s(M(C)) \leq R_s(M) \leq m_M$$

Proof: Consider an optimal history for R_s for the matrix M , i.e. a sequence of column deletions, row deletions and non-redundant row removal events which reduces a

matrix M to the empty matrix and requires $R_s(M)$ non-redundant row removal operations. This history can be used to obtain R_s histories for each connected component as follows. Consider any non-redundant row removal operation in the optimal history and let r denote the row removed. There is at least one connected component C such that the row r is non-redundant in the submatrix restricted to the sites in C . Let C' be another connected component in the conflict graph of M . We claim that the row r is redundant in the submatrix restricted to the sites in this component C' . If this was not the case, it would contradict Theorem 11. Hence, the row r is non-redundant in the submatrix restricted to the sites of exactly one connected component. Therefore, every non-redundant row removal can be assigned to the R_s history of one connected component. For all other components, this row removal corresponds to a row deletion event in the history. Therefore, the sequence of column deletion, row deletion and row removal operations in the history of a connected component is identical to that in the optimal R_s history for M . Moreover, the total number of non-redundant row removal operations summed over histories of all connected components is exactly $R_s(M)$. It follows that $\sum_{C \in \mathcal{CC}} R_s(M(C)) \leq R_s(M)$. Since $R_s(S) \geq R_h(S)$ for any set of sites S , the sum of R_h bounds on the connected components is also a valid lower bound. ♣

Note that computing R_s is intractable in general. However, we can possibly speed up the computation of the R_s bound by computing the bound independently on each connected component of the conflict graph. Moreover, in practice the above lemma can be used to obtain improved bounds as follows. For each connected component, it is easy to check in polynomial time if $R_s = 1$ or more. If it is not then it cannot be explained by a single recombination event. Hence, instead of one event, one can infer two recombination events. In fact for any small constant c , one can check if $R_s < c$ or not. We illustrate this on a real dataset in subsection 3.3.1. Moreover, the above lemma also allows us to combine the bounds R_h and R_c . For a given set of sites, one can get a bound that is at least as good as the maximum of the bounds R_h and R_c .

We can use lemma 3 in conjunction with the connected component lower bound to obtain a somewhat stronger lower bound on m_M . For a subset S' of the sites in M , let $\mathcal{CC}(S')$ denote the non-trivial connected components in the conflict graph for $M(S')$. For every matrix M ,

$$\max_{S' \subseteq S} |\mathcal{CC}(S')| \leq m_M \quad (3.1)$$

We introduce the Max-NTCC problem for finding a subset of sites whose conflict graph has the maximum number of non-trivial connected components. Unfortunately, we show that this problem is NP-complete.

Max-NTCC problem:

Input: A matrix M with n sequences and a set S of s sites.

Output: $S' \subseteq S$, such that the number of non-trivial connected components in the conflict graph of $M(S')$ is $\geq k$.

Theorem 13: The Max-NTCC problem is NP-complete.

Proof: It is easy to see that the problem is in NP. To prove the NP-hardness, we give a reduction from the Independent Set problem. The independent set problem is defined as follows: Given an undirected graph $G = (V, E)$, is there a subset V' of V of cardinality $\geq k$ such that there is no edge between any pair of vertices in V' .

We construct a matrix M with $2|V|$ sites (columns) and $3|V| + 3|E|$ rows as follows. Label the nodes in V arbitrarily from 1 to $|V|$. For every vertex v_i in V we define 2 sites v_i and v'_i . We initially start with no rows and add new rows to the matrix keeping the number of columns fixed as $2|V|$. For every vertex v_i , we add 3 rows with the pairs $\{01, 10, 11\}$ in the columns $\{v_i, v'_i\}$ and with value 0 in all other columns. Hence, we obtain a matrix with $3|V|$ rows, such that there is a conflict between the sites $\{v_i, v'_i\}$, $1 \leq i \leq |V|$ and no other conflicts. Now, for every edge $(v_i, v_j) \in E$, we add three new rows with the pairs $\{01, 10, 11\}$ in the columns $\{v_i, v_j\}$ and with value 0 in all other columns. As a result, we obtain a matrix M with $2|V|$ columns and $3|V| + 3|E|$

rows. We claim that the only edges in the conflict graph for this matrix are of the form $\{v_i, v'_i\}$, $v_i \in V$ or $\{v_i, v_j\}$, where $(v_i, v_j) \in E$. This is true since the only pairs of columns for which there is a row with pair $\{11\}$ are $\{v_i, v'_i\}$, $v_i \in V$ and $\{v_i, v_j\}$ where $(v_i, v_j) \in E$.

Suppose that there exists an independent set $V' \subseteq V$ of cardinality k in G . Consider the conflict graph for $M(S')$ where $S' = \cup_{u \in V'} \{u, u'\}$. Each pair of sites $\{u, u'\}$, $u \in V'$ forms a connected component of size 2, since there is no conflict between a pair of sites (u, v) where $u, v \in V'$. Hence the conflict graph $G_C(M(S'))$ has k non-trivial connected components.

Now, let $S' \subseteq S$ be such that the conflict graph for $M(S')$ has k non-trivial connected components. It is easy to see that every non-trivial connected component has at least one non-primed vertex $u \in V$. For each connected component, we choose one non-primed vertex to form the set $I \subseteq V$. Now, I is an independent set in G since if there was an edge between two vertices in I then they would have been in the same connected component in $\mathcal{CC}(M(S'))$ and therefore not both in the set I . Also, the independent set I has cardinality k . Hence, there is an independent set $V' \subseteq V$ of cardinality at least k iff there exists a subset S' of S such that the conflict graph of $M(S')$ has k non-trivial connected components. ♣

3.3 Comparison of R_c with other bounds

In this section, we compare R_c to the bounds R_m and R_h . We have already proved (see Theorem 10) that the history based bound R_s is always better than the bound R_c , however it is not feasible to compute R_s for a set of 10 or more haplotypes (see Myers and Griffiths (2003)). First, we observe that if we apply the R_c method to subsets of continuous columns, and compute the best bound using dynamic programming on the local lower bounds, then R_c is always better than R_m . Note that the running times for computing the bounds R_h and R_s are exponential and super-exponential respectively. In general, the best lower bound that can be obtained using the connected component

approach is $m/2$ where m is the number of columns. If the number of distinct haplotypes $n > m + m/2$, then the bound R_h is trivially better than the connected component lower bound. For example, for a set of 2^k haplotypes with k columns, the R_h bound is exponentially better than the connected component lower bound. For regions of low diversity in haplotype data, the connected component lower bound can possibly offer better bounds than the haplotype diversity bound R_h . Here, we provide an example of a matrix M for which $R_h = \frac{2}{3}|\mathcal{CC}|$, where \mathcal{CC} is the set of the non-trivial connected components in the conflict graph for M . Although this example is not real, it serves to illustrate the kind of haplotype data for which the bound R_c could offer improvements over the bound R_h .

Theorem 14: For all n_0 , there exists a matrix M with $n \geq n_0$ rows such that $R_h(M) = \frac{2}{3}R_c(M)$.

Proof: We choose the number of rows for the matrix M to be $3^k \geq n_0$, where $k \geq 2$. Starting from an empty matrix, we add new columns keeping the number of rows fixed. We add columns in groups of 2, which represent a connected component in the conflict graph. The following procedure defines the matrix M (depicted in Figure 3.2) :

1. **for** $j = 1$ to $k - 1$
2. **for** $i = 0$ to $3^{k-j} - 1$ **do**
3. add two new columns with the following values:
4. 01 in 3^{j-1} rows starting from row $3^{j-1}(3i + 1)$
5. 10 in 3^{j-1} rows starting from row $3^{j-1}(3i + 2)$
6. 11 in 3^{j-1} rows starting from row $3^{j-1}(3i + 3)$
7. 00 in the remaining rows

Claim: Every column conflicts with only one other column.

Proof: Consider a column i where $0 \leq i \leq 2 \cdot 3^{k-1} - 1$. There are only two rows with a 1 at this site and every other site (apart from the one site this site conflicts with) has the same value in these two rows. Hence, every site $i, 0 \leq i \leq 2 \cdot 3^{k-1} - 1$ is involved in

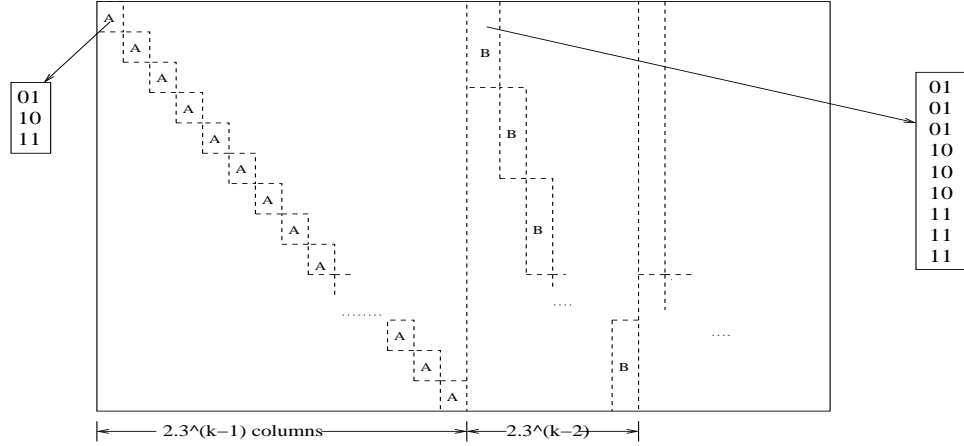


Figure 3.2: The structure of the matrix M for which $R_h = \frac{2}{3}R_c$

only one conflict. Next, consider a column j where $2 \cdot 3^{k-1} \leq j \leq 2 \cdot 3^{k-1} + 2 \cdot 3^{k-2} - 1$. There are six rows with a 1 at this site and every other site (except the column with which j conflicts) has the same value in these rows. Hence, every site j in the range $[2 \cdot 3^{k-1}, 2 \cdot 3^{k-1} - 1 + 2 \cdot 3^{k-2} - 1]$ is involved in only one conflict. Similar arguments are applicable to each submatrix added in steps 2-6 of the procedure above which describes the matrix M . Hence, the required property holds for every column in the matrix M .



Lemma 15: R_h for the matrix M (constructed above) is exactly $\frac{2}{3}\mathcal{CC}$.

Proof: After adding new rows as defined by this procedure, the matrix M has $2(3^{k-1} + 3^{k-2} + \dots + 3) = 3^k - 3$ columns. Also, there is a conflict between any two new columns added. Therefore, we have $\left\lfloor \frac{3^k - 3}{2} \right\rfloor$ non-trivial connected components. Hence $|\mathcal{CC}| = \left\lfloor \frac{3^k - 3}{2} \right\rfloor$.

Observe that the first two columns are the only columns that can distinguish rows 1, 2 and 3. The next 2 columns are the only columns that can distinguish rows 4, 5 and 6. In general, columns $2i$ and $2i + 1$ are the only columns that can distinguish between rows $3i$, $3i + 1$ and $3i + 2$, $0 \leq i \leq 3^{k-1} - 1$. Let I be the set of the first 3^{k-1} sites. Observe that $|D(M_I)| = 3^k$. Restricting the matrix to the first $2 \cdot 3^{k-1}$ sites, we obtain

$R_h \geq 3^k - 2 \cdot 3^{k-1} - 1 = 3^{k-1} - 1$. Now, we need to show that for every subset of rows S , $(|D(M_S)| - |S| - 1) \leq 3^{k-1} - 1$. Suppose on the contrary, there is a subset S for which $(|D(M_S)| - |S| - 1) > 3^{k-1} - 1$. If S does not contain a pair of columns $(2i, 2i + 1)$, $0 \leq i < 3k - 1$, then we can add the pair of columns S to obtain a set of columns S' such that $(|D(M'_S)| - |S'| - 1) > (|D(M_S)| - |S| - 1)$. In the other case, where S does not contain one of the columns $(2i, 2i + 1)$, we can add that column to get a set S' , for which $(|D(M'_S)| - |S'| - 1) = (|D(M_S)| - |S| - 1)$. Inductively, we can add columns to S to obtain a set of columns $S^* = S \cup I$ such that $(|D(M_{S^*})| - |S^*| - 1) \geq |D(M_S)| - |S| - 1 > 3^{k-1} - 1$. We know that $|S^*| \geq |I| = 2 \cdot 3^{k-1}$. Hence, we obtain $|D(M_{S^*})| > 3^{k-1} + 2 \cdot 3^{k-1} = 3^k$, which is a contradiction since we only have 3^k rows. Therefore, it follows that $R_h = 3^{k-1} - 1$. Hence,

$$R_h = 3^{k-1} - 1 = \frac{2}{3} \left(\frac{3^k - 3}{2} \right) = \frac{2}{3} |\mathcal{CC}|$$

This shows that $R_h = \frac{2}{3} R_c$ for the matrix M . For this particular example, one can also show that $R_s = |\mathcal{CC}|$. ♣

This completes the proof of Theorem 14. ♣

Although the R_h bound for the matrix M is $3^{k-1} - 1$, by obtaining local bounds using R_h on subregions of the matrix and using the framework of Myers and Griffiths (Myers and Griffiths, 2003) to combine these local bounds, the overall bound for the whole matrix can be improved to $|\mathcal{CC}|$. However, by permuting columns appropriately, the overall bound obtained by combining the local R_h bounds can be forced to be $3^{k-1} - 1$, while the connected component bound is unchanged. The next theorem shows that the above example is in fact a worse case scenario.

Theorem 16: For any matrix M , $R_h(M) \geq \frac{2}{3} R_c(M) - \frac{1}{3}$.

For a given matrix M , we can remove columns such that every non-trivial connected component is of size 2 and the number of non-trivial connected components does not decrease. Therefore, it suffices to prove the above theorem for a matrix M in which

every connected component has size 2. Next, we prove a series of lemmas for a matrix M in which every connected component is of size 2. For such a matrix M with n distinct rows, we show that the number of non-trivial connected components cannot exceed $n/2$. For a matrix with n rows, we denote the number of non-trivial connected components by $C(n)$. Since every component is of size 2, the number of sites is $2 \cdot C(n)$.

Lemma 17: For a matrix M in which every connected component is of size 2, there exists a connected component (pair of columns), such that 3 of the 4 pairs $\{(00, 01, 10, 11)\}$ appear exactly once.

Proof: Consider a connected component C . Let $C(ab)$ denote the set of rows with value ab in the columns of C , where $ab \in \{00, 01, 10, 11\}$. Let $C(2)$ denote the set of rows corresponding to the second largest among the four values: $\{|C(ab)| : ab \in \{00, 01, 10, 11\}\}$. and xy denote the pair in the component C in the rows $C(2)$. Let C_{min} be the connected component for which $|C_{min}(2)| = \min_{C \in \mathcal{CC}} \{|C(2)|\}$.

If $|C_{min}(2)| = 1$, then clearly $|C_{min}(2)| = |C_{min}(3)| = |C_{min}(4)| = 1$, and therefore 3 of the 4 pairs appear exactly once in the connected component C_{min} . If $|C_{min}(2)| > 1$, since all the rows in the matrix M are distinct, there exists a connected component C' such that the pairs in the component C' in the set of rows $C_{min}(2)$ are not all equal. Hence, $h(C_{min}, C') = xy$ (here $h(A, B)$ denotes the common haplotype of component A with respect to B) and therefore 3 out of 4 pairs in the component C' are present in the rows $C_{min}(2)$. Let $C'(2)$ denote the set of rows corresponding to the second largest among the four values: $\{|C'(ab)| : ab \in \{00, 01, 10, 11\}\}$. Clearly, $|C'(2)| < |C_{min}(2)|$. However, $|C_{min}(2)| = \min_{C \in \mathcal{CC}} \{|C(2)|\}$ which leads to a contradiction. Therefore, $|C_{min}(2)| = 1$ and there is a connected component such that 3 of the 4 pairs $\{00, 01, 10, 11\}$ appear exactly once.

♣

Lemma 18 For a matrix M in which every connected component is of size 2, $R_h(M) \geq n/3 - 1$.

Proof: The proof is by induction on the number of rows. For a matrix with at most 6 rows and at least one connected component(pair of columns), we can restrict the matrix to a single connected component for computing R_h and hence $R_h \geq 4 - 2 - 1 = 1 \geq 6/3 - 1$. This proves the base case. Suppose the induction hypothesis is true for $k < n$, i.e for every matrix M with k rows ($k < n$) and in which every connected component is of size 2, $R_h(M) \geq k/3 - 1$. Now, consider a matrix with n rows, where $n > 6$. From the previous lemma, there is a pair of conflicting columns (connected component), such that 3 of the 4 pairs appear exactly once. Let M' denote the matrix after removing the two columns with 3 of the 4 pairs occurring only once and the three rows corresponding to the three pairs. Note that removing the two columns does not cause any other rows to become identical, since all rows apart from the three removed had the same value in the two columns. (see lemma 6) One can write $R_h(M) \geq R_h(M') + 3 - 2 = R_h(M') + 1$. From the induction hypothesis, we have $R_h(M') \geq (n - 3)/3 - 1$. Combining, we obtain $R_h(M) \geq n/3 - 1$, which proves the lemma. ♣

Lemma 19: For a matrix M in which every connected component is of size 2, $C(n) \leq n/2 - 1$.

Proof: Consider a matrix M with n rows in which every connected component has size 2. From Lemma 17, there exists a connected component C , such that 3 of the 4 pairs occur exactly once in C . Denote the 3 rows containing these pairs as $R(C)$. Moreover, applying the 2-edge theorem, we also have the property that the pairs in the rows $R(C)$, in every component apart from C are identical. Hence, if we remove the columns in C , three rows in M become identical. Therefore, we have the equation: $C(n) \leq 1 + C(n - 2)$. We also have $C(4) = 1$ and hence $C(n) \leq n/2 - 1$. ♣

From the above two lemmas, it follows that $R_h \geq n/3 - 1 \geq \frac{2}{3}C(n) - \frac{1}{3}$, which completes the proof of Theorem 16. Note that the theorem still holds if $R_c(M)$ is replaced by $\max_{S' \subseteq S} R_c(M(S'))$.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
B	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
D	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
E	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1
F	0	1	0	0	0	1	0	0	0	1	0	1	0	1	1	1
G	0	1	0	0	0	1	0	0	1	1	1	1	1	1	0	1
H	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	1
I	1	1	1	1	0	1	0	0	1	1	1	1	1	1	0	1

Figure 3.3: A reduced haplotype matrix for alcohol dehydrogenase locus of *Drosophila*.

3.3.1 Application to a *Drosophila* Dataset

Next, we consider a real dataset taken from the alcohol dehydrogenase locus from 11 chromosomes of *Drosophila melanogaster* (Kreitman, 1983). The original dataset had 11 haplotypes and 2800 sites. We coalesce two identical haplotypes and remove all sites that are not incompatible with any other sites and sites that are identical to an adjacent site. This leaves us with a reduced set of 9 haplotypes typed at 16 sites (see Figure 3.3).

If the alleles at the sites 2 and 3 had not been determined, then the haplotypes would be restricted to the sites: $\{1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$. We denote a recombination lower bound for the sites between sites a and b by B_{ab} . For this smaller dataset, the recombination lower bound that we would get using R_m is 4. However, the conflict graph for the subset $\{1, 4, 5, 6\}$ has 2 connected components, which implies a local recombination lower bound of 2 between the sites 1 and 6, i.e. $B_{1,6} = 2$. Hence, the connected component lower bound for the smaller dataset is 5. However, for the set of sites between 7 and 15, there are two conflicting pair of sites: $(7, 15)$ and $(14, 15)$. One can check that the removal of one sequence does not destroy both these conflicts. Hence, one can infer a recombination bound of 2 for this subset (see Lemma 12), i.e. $B_{7,15} = 2$. Therefore, $B_{1,16} = B_{1,6} + B_{6,7} + B_{7,15} + B_{15,16} = 2 + 1 + 2 + 1 = 6$ which gives an overall lower bound of 6. For this dataset, Song and Hein (Song and Hein, 2004) showed that the minimum number of recombination events is 7. This example illustrates, that

if we have the alleles at fewer sites, then the connected component bound can provide improvements over the bound R_m .

3.4 Acknowledgement

Chapter 3, in full, was published in IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol 1(2), pp 78-90, 2004, V. Bafna and V. Bansal, “The Number of Recombination Events in a Sample History: Conflict Graph and Lower Bounds”. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Detecting large inversions from whole-genome SNP haplotype data

4.1 Introduction

Long before SNPs were discovered as a form of genetic variation and even before the discovery of DNA, Dobzhansky, in early 20th century, had detected large genomic rearrangements that were polymorphic in *Drosophila* strains. These were the first known examples of *genetic variation* that Darwin's theory had described as the 'raw materials for evolution'. Advancements in modern biology have allowed us to study and cataloger the smallest form of genetic variation (SNP). However, with sequencing of complete mammalian genomes, large chromosomal rearrangements have come under increasing attention with extensive work on their discovery, mechanisms of origin and impact on genomic evolution. Study of chromosomal rearrangements is also important for reasons of medical importance since genomes of tumor cells are known to undergo extensive rearrangements. These chromosomal rearrangements include inversions, duplications, translocations and deletions of genomic segments of sizes ranging from a gene to several megabases. Large scale structural changes such as deletions, duplications, inversions and translocations of genomic segments are known to be asso-

ciated with susceptibility to disease (Lakich et al., 1993; Osborne et al., 2001; Lupski, 1998). In the past 2-3 years, increasing evidence suggests that the human genome contains large scale DNA variants, collectively referred to as 'structural variants'. High-throughput experimental techniques based on comparative genomic hybridization have enabled the discovery of hundreds of copy number polymorphisms in human individuals (Sebat et al., 2004; Iafrate et al., 2004; Sharp et al., 2005). The HapMap genotype data has also been used to discover insertion/deletion polymorphisms (Conrad et al., 2006; Hinds et al., 2006; McCarroll et al., 2006).

In sharp contrast, knowledge about the location and genome-wide extent of inversion polymorphisms has not accumulated at the same pace, primarily due to the lack of a high-throughput technique for detecting inversions. Inversion polymorphisms are well known and quite common in species of *Drosophila*. Sturtevant discovered in 1921 (Sturtevant, 1921) that the genomes of two *Drosophila* species differ a large inversion on one of the chromosomes. Inversion polymorphisms were shown to reduce recombination rates in the inverted region and increase recombination rates in other chromosomes (SCHULTZ and REDFIELD, 1951). Recombination is suppressed in individuals who are heterozygous for the inversion: carry both the non-inverted and the inverted haplotype. This leads to an overall reduction in the recombination rate in the region. Lack of recombination between the two haplotypes causes them to evolve independently accumulating mutations that are specific to each clade. Large inversion polymorphisms are generally believed to be rare in humans due to their expected deleterious effects and very examples of such polymorphisms are known. A notable one is the recently discovered 900kb long common inversion polymorphism on 17q21.31 (Stefansson et al., 2005). The inverted orientation had a frequency of 21% in Europeans but was rare in individuals of African (6%) and Asian (1%) origin. The inverted haplotype was dated to be about 3 Myr old but shows little evidence for recombination, leading to a distinct haplotype pattern and extended LD across the region in the CEU population (see Figure 4.1 for a graphical display of the unusual haplotype pattern for this region). Interestingly, genotype-phenotype analysis in an Icelandic population showed

that women carrying the inverted haplotype had more children than those who didn't, providing direct evidence that the inverted arrangement is under some form of selection. Stefansson et. al. (Stefansson et al., 2005) suggested that an computational approach to detecting inversion polymorphisms would be to look for regions with unusually high LD especially long range LD. However, there is great variation in recombination rates across the human genome, and high LD can arise just to an underlying low recombination rate. There are patterns that one would expect to see in a region harboring an inversion polymorphism that are unlikely due to fluctuation in recombination rates and under neutral evolution.

Another large common inversion polymorphism discovered previously, lies in the chromosome band 8p23.1 - 8p22. This inversion was initially detected when unusual recombination patterns were observed in recombination analysis of CEPH pedigree data (Broman et al., 2003) and later verified and analyzed using Fluorescent *in situ* hybridization (FISH) (Giglio et al., 2001). Subsequently, the frequency of the inversion polymorphism was determined to be about 26% in the CEU population and 39% in the Japanese population (Sugawara et al., 2003) and the inversion breakpoints were mapped quite precisely. For this unusually long inverted region (4.7 MB), the reference assembly (Build 34) has the orientation of the minor allele while the ordering in the deCODE genetic map (Kong et al., 2002a) matches the major allele.

A recent study (Tuzun et al., 2005) mapped fosmid paired-end sequence data from a fosmid DNA library of a North American female (not represented in the reference human genome assembly) to the reference human assembly. Fosmids that showed discrepancy by size were indicative of deletions/insertions between the two genomes, while fosmids whose ends mapped on the same strand of the reference genome (discordant by orientation) pointed to potential inversions. This strategy revealed 56 putative inversion breakpoints in addition to 139 insertions and 102 deletions. For 228 of the 297 variants, the fosmid library also contained clones consistent with the reference assembly, suggesting that these are unlikely to be genome assembly errors. Although the method is effective in determining inversions, it will require extensive re-sequencing in a popula-

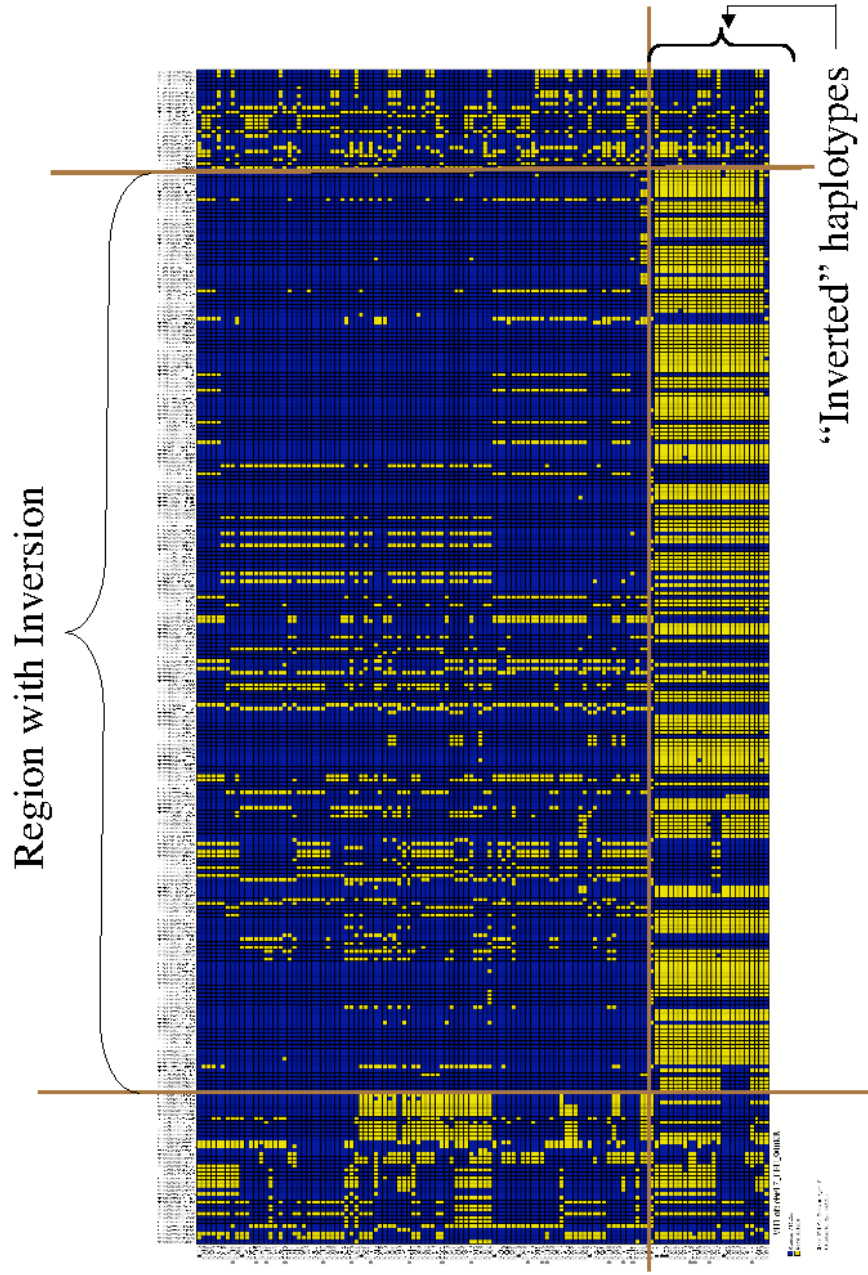


Figure 4.1: Haplotype patterns for the 900kb long inversion on chromosome 17 in the CEU HapMap sample. Each row corresponds to a chromosome and each column is a SNP. The haplotypes are clustered based on similarity. The inverted haplotypes are labeled and the inversion boundaries are marked by vertical bars. (Figure generated using program <http://pga.gs.washington.edu/VH1.html>)

tion of individuals to fully determine the extent and frequency of these polymorphisms. An indirect approach that has been adopted for finding inversion polymorphisms is to test human-chimp inversions for polymorphism in humans using FISH and PCR analysis (Feuk et al., 2005; Szamalek et al., 2006). Out of 23 regions that were tested (Feuk et al., 2005), 3 were found to be polymorphic in humans with the largest being a 800kb inversion on chromosome 7.

In this chapter, we describe a statistical method to detect large inversion polymorphisms in the human genome using whole genome SNP genotype data. Unlike deletions which cause miscalled genotypes and can lead to Mendelian inconsistencies (McCarroll et al., 2006; Conrad et al., 2006), inversions are copy neutral and do not affect the SNP genotypes. Our method is based on the detection of an unusual Linkage Disequilibrium pattern that is indicative of inversions for which the inverted orientation (w.r.t reference human genome sequence) is present in a majority of chromosomes in a population. The method can also detect assembly orientation errors in the human sequence assembly, i.e. genomic segments which are present in the reverse orientation in the assembly. Using simulations, we show that our method has statistical power to detect such inversions. We have applied our method to data from the first phase of the International HapMap project to generate a list of 176 candidate inversions in the three HapMap ‘analysis panels’ (CEU, YRI and CHB+JPT). Although it is difficult to estimate how many of these represent true inversions, a crude estimate of the false positive rate using coalescent simulations indicates that about half of the 78 predictions in the YRI ‘analysis panel’ represent true inversions. The false positive rate could be higher (about 80%) for the inversions in the CHB+JPT ‘analysis panel’, according to a conservative assessment. Even with the high false positive rates, our method is a cost-effective approach to discovering inversion polymorphisms. We have looked for supporting evidence for our predicted inversions in the form of discordant fosmid pairs, assembly discrepancies and presence of a pair of inverted repeats near inversion breakpoints. This has resulted in a smaller list of 15 inversions, two of which represent previously known inversions.

4.2 Methods

Genetic maps are constructed by genotyping a large number of genetic markers in a pedigree and ordering the markers based on estimates of the recombination fraction between genetic markers. Markers that are physically close to each other in the genome are also expected to be close to each other in the recombination map and vice versa. On the other hand, the human genome assembly represents the genomic sequence of a few individual(s) and a second, possibly different, ordering of the markers (SNPs for example) can be determined by mapping the sequence flanking the markers to this reference. Recently a high resolution genetic map was constructed using data for an Icelandic population (Kong et al., 2002a). Comparison of the genetic map to the reference sequence revealed several regions where the ordering of the genetic markers was in opposite orientation to that suggested by the reference sequence. Given the incomplete nature of the draft human sequence at that time, the sequence was modified in the regions where the genetic map strongly indicated a different marker order. The possibility that some of these discrepancies are a result of an inversion polymorphism in the particular region cannot be discounted. For example, if the human sequence represents the minor allele in a particular region of the human genome which has two orientations, one would expect the ordering of the markers (inside the inverted segment) in the genetic map to be consistent with that of the major allele and hence be opposite to that of the sequence. In fact, this is true for a 4.5 megabase long inversion on chromosome 8 where the reference human sequence represents the minor allele (frequency 20-30% in human populations) and the genetic map (Kong et al., 2002a) matches the marker order of the major allele. However, the low resolution of genetic maps makes it difficult to detect such discrepancies in general.

In genotype data from unrelated individuals, *Linkage Disequilibrium (LD)* refers to the non-random association of alleles at physically neighbouring markers (SNPs in our case). In human population data, significant LD is observed at close distances and little or no LD is observed at long distances. This correlation of LD with distance is

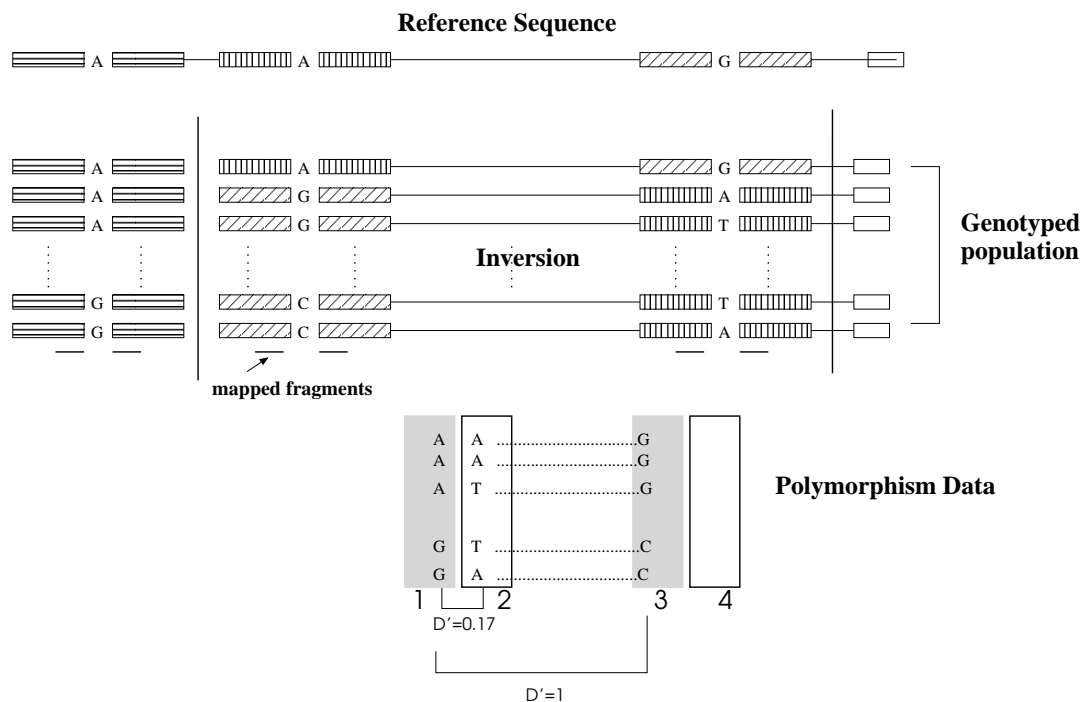


Figure 4.2: Unusual Linkage Disequilibrium observed in SNP data when the inverted haplotype (w.r.t the reference sequence) has very high frequency. SNPs are 'mapped' to the reference sequence using the flanking sequence (denoted by shaded boxes). Therefore close SNPs in high LD are mapped to distant regions 1 and 3 (the shaded boxes). Consequently, the two regions show unusually high LD for that distance.

very noisy due to multiple factors, largely due to the fine-scale heterogeneity in the recombination rates in the human genome (McVean et al., 2004; Crawford et al., 2004; Myers et al., 2005). Although it may not be possible to determine a physical ordering of SNPs using LD alone, it is possible to distinguish between SNPs that are physically close from physically distant SNPs using LD. Our method utilizes high density SNP haplotype data to find regions of the human genome where the ordering of the SNPs suggested by Linkage Disequilibrium patterns is opposite to that of the physical sequence.

4.2.1 Using LD in population data to detect inversions

Consider a genomic region that is inverted (w.r.t the reference sequence) in a majority of the chromosomes in a population and assume that we have genotyped mark-

ers on either side of the two breakpoints. For a graphical illustration, see Figure 4.2. In such a scenario, we would expect to see unusually high long range LD (LD_{13} and LD_{24}) than would be expected between markers that are physically distant. Further, one would also observe low LD (LD_{12} and LD_{34}) between pairs of markers that are physically close according to the reference sequence. The strength of this effect will be proportional to the frequency of the inverted allele. Our statistic is designed to search for pair of breakpoints showing this kind of signal.

4.2.2 The Inversion Statistic

Consider a pair of breakpoints where B_1 and B_2 denote two blocks on either side of the left inversion breakpoint and B_3 and B_4 are the blocks of SNP's spanning the other inversion breakpoint (See Figure 4.2). We compute a pair of log likelihood ratios, one for each inversion breakpoint which represent the log of the ratio of the probability of the region being inverted in the population vs being non-inverted. Let LD_{ij} denote the LD between blocks i and j , and d_{ij} denote the corresponding distance. The log likelihood ratio for the left breakpoint is defined as

$$LLR_l = \log \left(\frac{\phi_{d_{13}}(LD_{12}) \cdot \phi_{d_{12}}(LD_{13})}{\phi_{d_{12}}(LD_{12}) \cdot \phi_{d_{13}}(LD_{13})} \right) \quad (4.1)$$

Similarly, the log likelihood ratio for the right inversion breakpoint is defined as

$$LLR_r = \log \left(\frac{\phi_{d_{24}}(LD_{34}) \cdot \phi_{d_{34}}(LD_{24})}{\phi_{d_{34}}(LD_{34}) \cdot \phi_{d_{24}}(LD_{24})} \right) \quad (4.2)$$

If the pair of breakpoints represent inversion breakpoints (with the inverted allele having high frequency), we would expect the long range LD (LD_{13} and LD_{24}) to be stronger than the short range LD (LD_{12} and LD_{34}) and both log likelihood ratios to be positive. However, most measures of LD, including D' show some dependence upon allele frequencies. Therefore, even in the absence of an inversion, the log likelihood ratios could be positive (due to the long range LD being larger in magnitude than the short range LD just by chance). Therefore, we estimate the significance of the two log-likelihood ratios using a permutation test. For a pair of breakpoints denoted by (l_1, l_2)

and (r_1, r_2) , we permute the haplotypes inside the inverted region (from the block l_2 to r_1). The two log-likelihood ratios are computed for this permutation and the p -value is defined as the fraction of permutations for which at least one of the two log-likelihood ratios is greater than its corresponding original value. We use 10,000 permutations to compute each p -value. Using simulations, we found the p -value to have much better specificity and almost equal sensitivity at detecting inversions as compared to the log-likelihood ratios. Therefore, we use the p -value for a pair of breakpoints as our statistic for the presence of an inversion. The p -value for the log-likelihood ratios cannot be interpreted as a typical p -value; it estimates the chance that at least one of the two log-likelihood ratios would achieve the corresponding computed value even if there was no LD between the blocks.

4.2.3 Measuring LD

Most measures of LD are defined for a pair of bi-allelic sites, and have high variance. We are interested in assessing the strength of association between blocks of SNP's across the inversion breakpoints. Therefore, we use the *multi-allelic* version of the LD measure D' (Lewontin, 1964; Hedrick, 1987b) by considering a block of SNPs as a multi-allelic marker. Let A and B denote two blocks with haplotypes A_1, A_2, \dots and B_1, B_2, \dots respectively. Let p_i (q_j) denote the frequency of haplotype A_i (B_j). Define $D_{ij} = h_{ij} - p_i q_j$ where h_{ij} is the frequency of the haplotype $A_i B_j$. The extent of LD between each pair of haplotypes is defined as

$$D'_{ij} = \frac{D_{ij}}{D_{max}}$$

where

$$D_{max} = \begin{cases} \min\{p_i q_j, (1 - p_i)(1 - q_j)\} & \text{if } D_{ij} < 0 \\ \min\{p_i(1 - q_j), (1 - p_i)q_j\} & \text{otherwise} \end{cases}$$

The overall measure of LD between A and B is

$$D'_{AB} = \sum_i \sum_j p_i q_j |D'_{ij}|$$

We computed the LD measure between all pairs of multi-SNP markers on a chromosome (defined above) within a certain maximum distance. Using these LD values for the 22 autosomes, we obtained probability distribution curves of LD at a fixed distance d , denoted as ϕ_d . The X chromosome was excluded since it has a reduced recombination rate as compared to the autosomes.

4.2.4 Defining multi-SNP markers

For each ‘analysis panel’, all SNPs with a minor allele frequency smaller than 0.1 in the ‘analysis panel’ were discarded since they are less informative about LD patterns. After this filtering, we selected a multi-marker SNP block for every remaining SNP as follows. For each SNP S , we considered all SNPs in the genomic region $L(S) \dots L(S) + W$ where $L(S)$ is the genomic location of SNP S and W is the window size. If this window had less than k SNPs, it was discarded. For any k SNPs, an individual sequence is described by a haplotype of length k , induced by the allelic values of the k SNPs. Denote the set of haplotypes as A_1, A_2, \dots , with frequencies p_1, p_2, \dots respectively. For each window, we chose a subset of k SNPs that maximize the entropy of the haplotypes ($-\sum_i p_i \log p_i$) defined by any subset of k SNPs. The subset of SNPs with maximum entropy best captures the haplotype diversity of the window and is potentially most effective for measuring LD with other multi-allelic SNP markers. These k SNPs defined a multi-SNP marker with a left and right physical boundary defined by the physical location of the first and k th SNP. The average SNP density of the HapMap ‘analysis panels’ is about one SNP (with MAF ≥ 0.1) per 5-6 kb (across different chromosomes). The parameters k and W were chosen to be 3 and 18kb respectively based on this SNP density. The results are not greatly affected by increasing or decreasing W by a few kb. Simulations indicate that the power to detect inversions is smaller for $k = 4$ as compared to $k = 3$.

4.2.5 Simulating Inversions

It is straightforward to simulate an inversion with frequency $f = 1$, however, to the best of our knowledge, there is no existing program that can simulate human population data accommodating polymorphisms. The effect of decreasing the frequency of the inverted haplotype (on our statistic) is to essentially decrease the strength of long range LD and increase short range LD. Hence, we adopted a simple simulation strategy which could mimic this effect of the inversion frequency on our statistic directly. For a given chromosome, we chose at random two SNPs S and E that define the region with the inversion polymorphism. Let $1, 2 \dots s$ denote the SNP's in this chosen region. To simulate an inversion with frequency $f = 1$, we just flip the values of the alleles at SNPs i and $s + 1 - i$, for all $1 \leq i \leq s/2$ for all haplotypes. In order to simulate an inversion of frequency f ($0 < f < 1$), we randomly select a subset of haplotypes of size $f \times n$, where n is the total number of haplotypes. For every haplotype in this set, we simply flip the values of the alleles at SNPs i and $s + 1 - i$, for all $1 \leq i \leq s/2$. Notice that this may have the effect of combining the alleles at two different SNPs.

We used the phased haplotype data from the International HapMap project to simulate inversions. In order to simulate an inversion of given length, we choose one breakpoint randomly and the second breakpoint using the length of the inversion. After planting the inversions, we scan the chromosome for regions with low p -value for the log-likelihood ratios. A simulation inversion is considered to be detected if predicted inversion (l_1, l_2, r_1, r_2) has the property that the interval (l_1, l_2) overlaps the left endpoint of the inversion and (r_1, r_2) overlaps the right endpoint. Power is defined as the fraction of simulated inversions which are detected. Each point in the power plots is based on simulating about 500 inversions.

4.3 Results

4.3.1 Power to detect Inversion Polymorphisms

Our statistic is suited to detect long inversions (long enough for little or no long range LD to be present) for which the inverted orientation (w.r.t. the reference sequence) is the major allele. Many factors influence the power of our statistic, including background recombination rates, the length of the inversion, and the frequency of the inverted haplotype. We used simulations to assess how these factors affect the power of our statistic. Currently, only a few instances of inversion polymorphisms are known, and existing work on simulating population data incorporating the effect of inversion polymorphisms is of a theoretical nature based on *Drosophila* inversion polymorphisms (Navarro et al., 2000a). Therefore, we adopted a simple strategy to simulate inversions of varying frequency using haplotype data from the HapMap project.

As our simulations were over real data with high variation in recombination rates, we effectively average over the effect of recombination rate variation. Figure 4.3(a) describes the power of the statistic to detect inversions as a function of the frequency of the inverted allele (f), keeping the length fixed at 500 kb for the three HapMap ‘analysis panels’. The power is measured by the fraction of simulated inversions in which the inverted region was detected with a p -value less than a fixed cutoff (0.02). Figure 4.3(b) describes the power for different lengths of the inverted region. The results indicate that the power of the method is low for small inversions (0.45 for inversions of length 100kb) and increases with increasing length, saturating around 500kb. Although the simulations cannot completely capture the effect of an inversion on LD patterns, they suggest that our method has good statistical power to detect long inversions segregating at high frequency in a population. They also indicate that the power is maximum in the YRI ‘analysis panel’ (see Figure 4.3(a)). We show later, through independent assessment of the false-positive rate of our predicted inversions on the HapMap data, that the error rate is lowest for the YRI ‘analysis panel’.

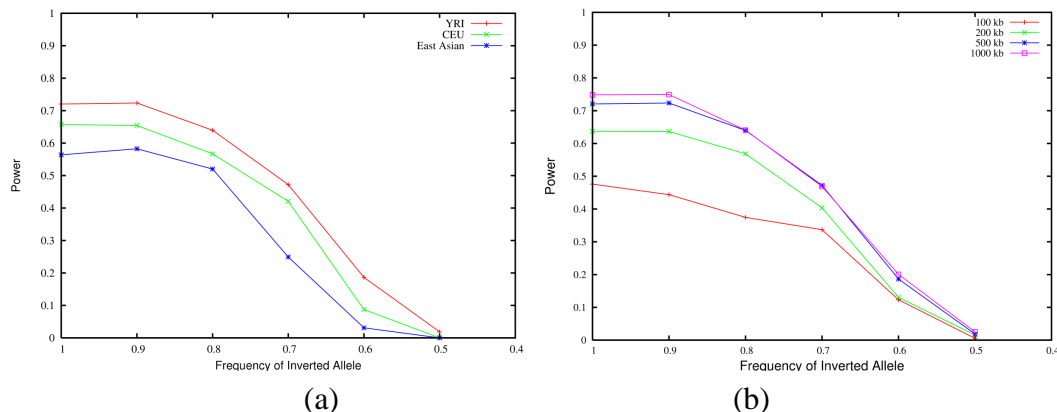


Figure 4.3: (a) Power of the method to detect inversion polymorphisms in the three HapMap 'analysis panels'. Inversions of varying frequency (100% to 50%) of a fixed length (500 kb) were simulated using the HapMap data for the three 'analysis panels' separately (YRI, CEU and CHB+JPT). The y-axis represents the fraction of simulated inversions for which there was at least one pair of predicted breakpoints with p -value ≤ 0.02 matching the breakpoints of the simulated inversion. (b) Power to detect inversions of four different lengths in the YRI 'analysis panel'.

4.3.2 Scanning the HapMap data for inversion polymorphisms

We utilized the genome-wide SNP data from Phase I of the International HapMap project consisting of genotypes of 269 DNA samples for approximately 1 million SNPs. These samples consist of 90 CEPH individuals (30 parent-child trios) from Utah, USA (abbreviated as CEU), 90 Yoruban individuals (30 trios) from Ibadan, Nigeria (YRI), 44 unrelated individuals from Tokyo, Japan (JPT) and 45 Han Chinese individuals from Beijing, China (CHB). We combined the individuals from the JPT and CHB populations to obtain a larger set of 89 individuals (referred to as the CHB+JPT 'analysis panel'). For the CEU and YRI 'analysis panels', our data consisted of 120 chromosomes (from the 60 parent individuals) each. We used the phased haplotype data (downloaded from the HapMap website) which was computationally phased using the program Phase 2 (The International HapMap Consortium, 2005; Stephens and Scheet, 2005).

We searched the phased haplotype data from the three HapMap 'analysis panel' individually using our statistic to determine sites of inversion. To reduce the number of

false positives, we considered predicted inversions with length in the range 200kb-4Mb. After clustering and filtering the initial list of predicted inversions for each ‘analysis panel’ separately (see Appendix for details of the clustering method), we had a total of 176 putative inversions in the three HapMap ‘analysis panels’ with a p -value of 0.02 or less. Of these, 26 were detected in the CEU ‘analysis panel’, 78 in the YRI ‘analysis panel’ and 72 in the CHB+JPT ‘analysis panel’. Most of the predicted inversions were unique to one of the ‘analysis panels’, but three regions were predicted in two ‘analysis panels’ each. The predicted list includes two sites of known inversion polymorphisms: a 800kb inversion polymorphism on 7p22.1 and a 1.1 megabase long inversion on chromosome 16p12.2. The 800 kb inversion at 7p22 was identified previously (Feuk et al., 2005) using interphase FISH with 2/20 CEPH individuals found to be heterozygous for the inversion. Our method gave a signal for this region in the YRI ‘analysis panel’ matching the known breakpoints (p -value of 0.012). For this inversion, the breakpoints were previously identified to a resolution of about 200kb (Feuk et al., 2005). For one of the breakpoints, our method can narrow down the location to a region of length 45kb. The chromosome 16 inversion was identified through the analysis of discordant fosmid pairs (Tuzun et al., 2005). Interestingly, we detected this inversion in both CEU (p -value 0.008) and the YRI ‘analysis panels’ (p -value 0.018) with identical pair of breakpoints (see Table 4.1). Analysis of the sequence around the breakpoints revealed that presence of a pair of long highly homologous inverted repeats (see Figure 4.4).

The current list of inversion polymorphisms in the human genome is small, with only about 15 inversions larger than 200kb that are known to be polymorphic in normal humans (from the Genome Variation Database at <http://projects.tcag.ca/variation/>). We looked for additional evidence that would support some of our predicted inversions. As noted earlier, sequence from different individuals (in the form of fosmid end pair sequences) can be mapped to the reference sequence to identify inverted regions (Tuzun et al., 2005). Another source of evidence comes from comparing the two human sequence assemblies produced by the International Human Genome Sequencing Consortium (International Human Genome Sequencing Consortium, 2001) and Celera

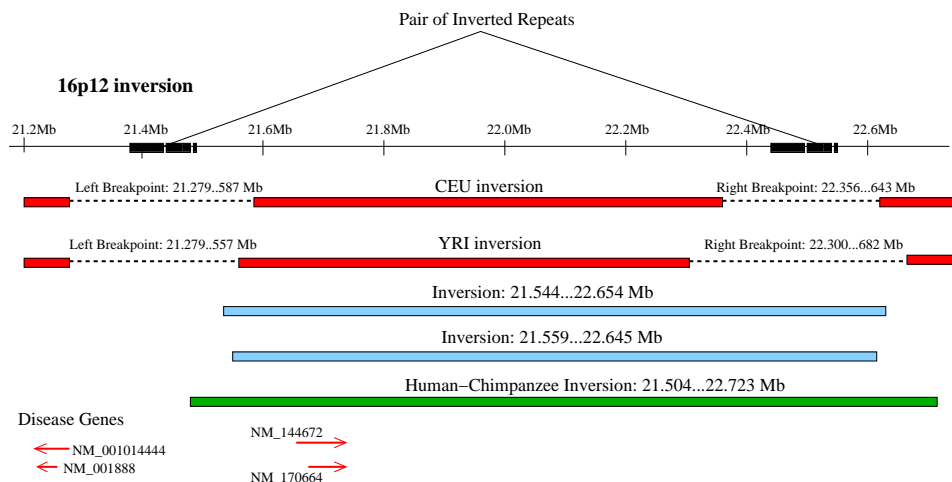


Figure 4.4: Genomic overview of a 1.4 Mb region at 16p12 predicted to have an inversion in both the CEU and YRI ‘analysis panels’. The left predicted breakpoint (the dotted line) overlaps with a ≈ 80 kb long segment that is highly homologous to a segment (in the inverted orientation) near the other breakpoint. The region contains several disease-related genes (from the OMIM database).

Genomics (Venter et al., 2001). Regions that are inverted in orientation between the two assemblies represent sites of assembly error in one of the two assemblies or polymorphic inversions, since these assemblies were generated using different sets of individuals. The Celera whole genome shotgun assembly (Istrail et al., 2004) was aligned to the reference sequence assembly (Build 34) to discover such regions (B. Walenz, pers. comm.). If the orientation of the Celera assembly supports a predicted inversion, then it is highly likely that the inverted orientation is present in the population.

One of our predictions was supported by two fosmid pair sequences discordant by orientation (Tuzun et al., 2005). This ≈ 1.2 Mb inversion on chromosome 10 (p15.1-p14) was predicted in the CHB+JPT ‘analysis panel’ with a p -value of 0.005. The left end of the fosmid pair mapped in the reference assembly about 40kb before the predicted left breakpoint while the right end mapped just before the right breakpoint (see Figure 4.5). Since the insert size of fosmids ranges between 32 and 48 kb, the two discordant fosmids are consistent with the predicted breakpoints. There were no gaps in the genome assembly near the breakpoints and there were fosmids and BACs consistent

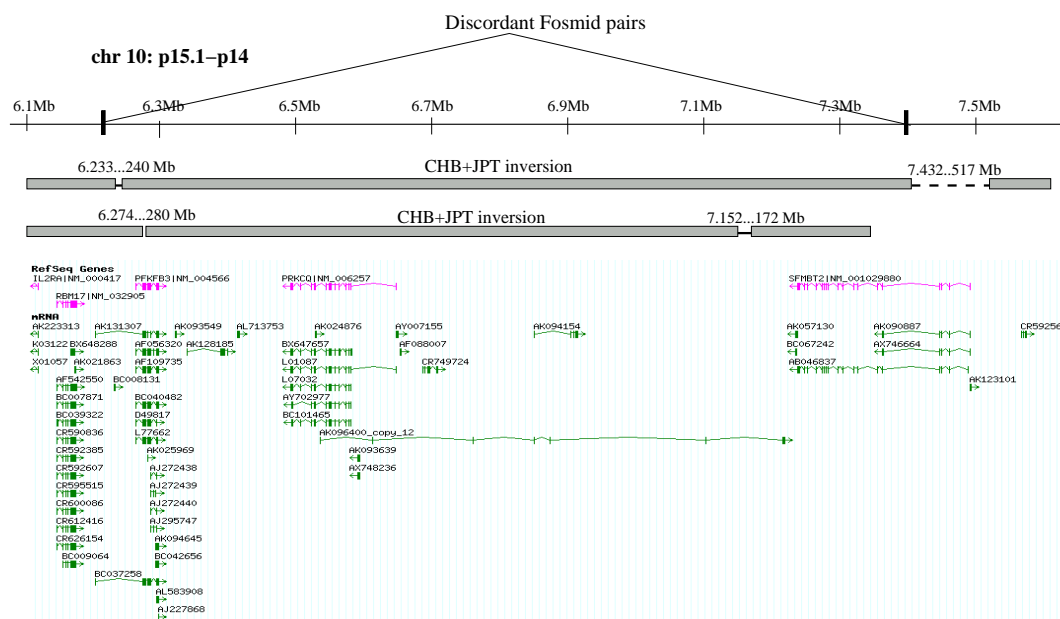


Figure 4.5: Overview of a ≈ 1.2 Mb long inversion on chromosome 10 predicted in the CHB+JPT ‘analysis panel’. Also shown are two fosmid pairs (discordant by orientation) whose one end maps to before the predicted left breakpoint and the other end maps to a region before the right breakpoint. These discordant mappings support the predicted inversion breakpoints. In this region, there is another overlapping inversion predicted in the CHB+JPT ‘analysis panel’. The region has several genes proximal to the left breakpoint, one of which is known to be over-expressed in tumor cells (Sampath et al., 2003).

with the reference assembly (UCSC Human Genome Browser: <http://genome.ucsc.edu>). This suggests that the inversion represents a previously unknown inversion polymorphism.

There were two regions for which we obtained evidence for the inverted orientation from the Celera assembly. One of these regions is a ~ 200 kb long region on chromosome 13 which was predicted to be inverted in both the CEU and CHB+JPT ‘analysis panels’. The region is also present in the inverted orientation in the Celera assembly and both breakpoints span large gaps (100kb) in the sequence assembly. Another large predicted inversion on chromosome 2p25 overlaps with a 1.4Mb region that is inverted between the two recent human genome assemblies (Build 34 and 35). The orientation of the Celera assembly of the human genome is concordant with the Build 35 assembly for the 1.4Mb region. There are gaps on each breakpoint which are not spanned by fosmids indicating that it is difficult to determine the correct orientation. This region was tested for polymorphism in a ‘analysis panel’ of 10 CEPH individuals (Feuk et al., 2005) but was not found to be polymorphic.

A 2Mb long predicted inversion on chromosome 10q.11 was predicted in both the YRI and CHB+JPT ‘analysis panels’. Further, both the breakpoints for this region span gaps in the human sequence assembly suggesting that this could represent an assembly orientation error. Two segments in this region are inverted between the Celera sequence assembly and the public assembly. The analysis of the genomic sequence around the breakpoints revealed the presence of several hundred kb long inverted repeats of very high sequence similarity.

Many of our predicted inversions overlap with regions that are inverted between the human and chimpanzee genomes (Newman et al., 2005; Feuk et al., 2005) (see Table 4.1 for a list). One of these is the 800kb inversion on chromosome 7 that was tested for polymorphism in humans since it was found to be inverted between the human and chimpanzee sequences (Feuk et al., 2005).

4.3.3 Sequence Analysis of Inversion Breakpoints

Segmental duplications have been shown to be highly overrepresented near sites of structural variation in the human genome (Iafrate et al., 2004; Tuzun et al., 2005). Mechanisms have been proposed as to how a pair of low copy inverted repeats may mediate inversion events in the genome (Giglio et al., 2001; Lupski, 1998; Shaw and Lupski, 2004). Pairs of inverted repeats have also been detected near the inversion breakpoints for several known inversion polymorphisms (Sugawara et al., 2003; Feuk et al., 2005). We checked for the presence of pairs of low-copy homologous repeats near the breakpoints of our predicted inversions. We found that 18 of our predicted inversions had pairs of highly homologous repetitive sequences near the breakpoints. There were 11 distinct regions for which there were inverted repeats near the breakpoints¹ (listed in Table 4.1). The significance of finding inverted repeats near the inversion breakpoints was estimated using a simple empirical method (see Appendix for details). The *p*-value was estimated to be 0.006.

Many examples of apparently benign chromosomal deletions that in many cases delete entire genes have recently been reported in the HapMap ‘analysis panels’ (Conrad et al., 2006; McCarroll et al., 2006). Less is known about inversions affecting genes by truncating the coding sequence in normal human individuals. Recurrent inversions disrupting the factor VIII gene on the X chromosome are known to be a common cause of severe hemophilia A (Lakich et al., 1993; Deutz-Terlouw et al., 1995; Bagnall et al., 2002). We analyzed the sequence around inversion breakpoints to see if they overlap with known genes in the human genome. The resolution of our predicted inversion breakpoints varies from a few kilobases in some cases to several hundred kilobases in others, making it difficult to say with certainty whether the inversion actually affects some gene. Assuming that purifying selection acts on inversions disrupting genes, one would expect a under-representation of inversion breakpoints disrupting genes. We found that 66 of our predicted inversion breakpoints are completely covered by one or

¹Some of these regions correspond to two predicted inversions

Table 4.1: List of predicted inversions for which there is some form of evidence supporting the inverted orientation. All genomic coordinates are based on Build 34 of the human genome assembly. Human-Chimp inversions are regions that are inverted between the human and chimpanzee genomes (Newman et al., 2005; Feuk et al., 2005). Inverted repeats imply the presence of a pair of low-copy highly homologous sequences, one near each breakpoint.

Chromosome	Left breakpoint (kb)	Right breakpoint (kb)	Analysis panel	p-value	Direct evidence	Inverted repeats	Human-chimp inversions
16	21,279 ... 587	22,356 ... 643	CEU	0.008	Inversion: 21,544 ... 22,654	✓	21,504 ... 22,723
	21,279 ... 557	22,300 ... 682	YRI	0.018	Inversion: 21,559 ... 22,645		
7	5,610 ... 783	6,632 ... 677	YRI	0.012	Inversion polymorphism Left breakpoint: 5,608 ... 776 Right breakpoint: 6,495 ... 735 Two discordant fosmid pairs	✓	5,766 ... 6,565
10	6,233 ... 240	7,432 ... 517	CHB + JPT	0.006			
10	46,378 ... 457	48,046 ... 735	YRI	0.014			
	45,506 ... 6,453	48,029 ... 821	CHB + JPT	0.018			
13	112,272 ... 388	112,556 ... 677	CEU	0.0002	112,373 ... 558 inverted between Build 34 & Celera		46,512 ... 47,057
2	112,266 ... 379	112,554 ... 665	CHB + JPT	0.005	Both breakpoints span gaps		
	1,527 ... 654	4,565 ... 681	YRI	0.005	1,627 ... 3,044 inverted between Build 34 & Celera; both break points span gaps		1,527 ... 3,040
1	143,737 ... 778	146,942 ... 7,113	YRI	0.005		✓	143,185 ... 143,723 143,862 ... 145,914 142,424 ... 146,586 131,015 ... 132,518 130,908 ... 132,285 175,531 ... 177,204 177,301 ... 532 149,120 ... 153,113 87,810 ... 878
2	132,383 ... 388	132,629 ... 654	YRI	0.005			
5	177,155 ... 499	180,364 ... 571	YRI	0.015			
7	148,971 ... 9,270	152,105 ... 161	YRI	0.007		✓	
9	87,120 ... 291	87,772 ... 886	CHB + JPT	0.0167		✓	
11	48,607 ... 841	50,765 ... 51,208	YRI	0.007		✓	49,337 ... 793 49,831 ... 49,871
12	124,704 ... 711	126,031 ... 044	CEU	0.0092		✓	
19	19,826 ... 871	20,331 ... 356	YRI	0.016		✓	
19	49,122 ... 128	49,564 ... 605	YRI	0.016		✓	

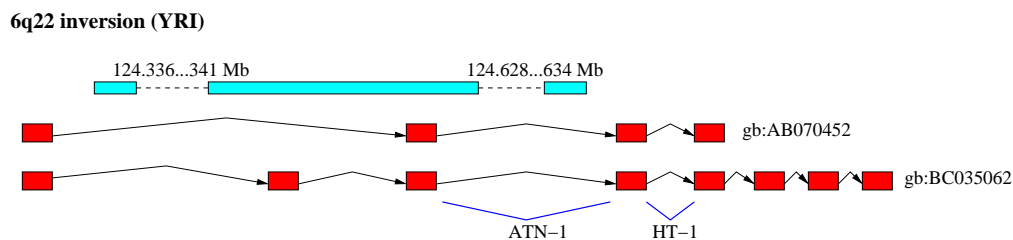


Figure 4.6: A predicted inversion on chromosome 6 (YRI sample) overlaps with the TCBA1 gene. The dashed line describes the location of the predicted breakpoints. The previously mapped breakpoints of the gene in T-cell lymphoma/leukemia cell lines are shown by the blue lines.

more genes (for 6 inversions, both breakpoints are spanned by gene(s)). This is significantly less than what one would expect by chance (p -value of 0.02).

Many of the genes that intersect with breakpoints are previously known to be disrupted in diseases. The T-cell lymphoma breakpoint-associated target 1 (*TCBA1*) gene spans a genomic region of over 900kb on chromosome 6, and is associated with multiple splice isoforms, as well as alternative start sites. As the name suggests, the gene is structurally disrupted in T-cell lymphoma cell lines (Tagawa et al., 2002), and developmental disorders (Yue et al., 2006). A sketch of the previously mapped breakpoints and our predicted inversion breakpoints with respect to the known isoforms of the gene is shown in Figure 4.6.

We also detect a number of disrupted genes with alternative splice forms, with some of the splice isoforms consistent with the inversion breakpoint. An interesting example is the Islet cell antigen (ICAp69) gene, which is a target self-antigen in type 1 diabetes. The gene is known to have multiple isoforms (Gaedigk et al., 1996). As shown in Figure 4.7, a predicted inversion breakpoint on chromosome 7 removes the 3' end of the gene (gb:BC008640), approximately consistent with the expression of alternative splice forms (gb:BC005922,U38260). These and many other examples hint at the important role of structural variation in mediating gene diversity.

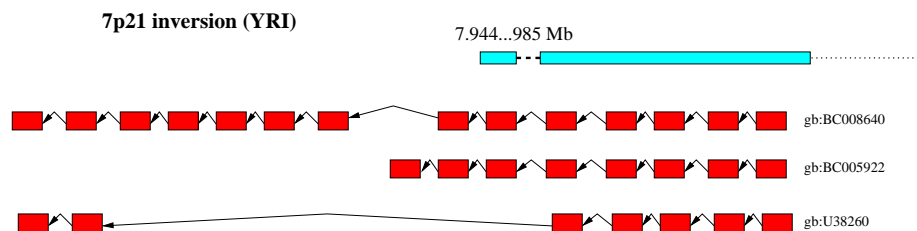


Figure 4.7: Splice isoforms of the ICAp69 gene that are approximately consistent with a predicted YRI inversion breakpoint on chromosome 7. The region of the left insertion breakpoint is denoted by a dashed line. The exons are not drawn to scale.

4.3.4 Assessing the false positive rate

Several of our predicted inversions represent known inversion polymorphisms and many others are supported by independent forms of evidence such as matching fosmid end sequences showing discordancy by orientation, regions inverted between different human assemblies, etc. Given the incomplete nature of our knowledge of inversion polymorphisms in the human genome, this does suggest that many of our other top predictions could represent inversions. Although LD generally decays with increasing distance between the markers, it is now well known that there is significant variation in recombination rates across the human genome (McVean et al., 2004; Myers et al., 2005). This variation in the recombination rates could potentially result in false positives using our statistic. Therefore, it is useful to estimate how many of our predicted inversions are correct. Estimating the false positive rate reliably is difficult, given the state of our knowledge.

We used coalescent simulations to estimate the frequency of predicted inversions on haplotype data with ‘no inversions’. To incorporate the heterogeneity in recombination rates in the simulated data, we used a recently developed coalescent simulation program (Schaffner et al., 2005) which can generate population data incorporating variation in recombination rates and a wide range of demographic histories for different populations (see Appendix for details of the coalescent simulations). The program is calibrated to produce haplotype data that has considerable variation in LD such as that

seen in real population data. The same thresholds and parameters were used for scanning the simulated datasets using our statistic as for the HapMap data. We analyzed the number of predicted inversions in the simulated data separately for each ‘analysis panel’ . Given the small number of predicted inversions in the HapMap data and many caveats in matching the simulation parameters with the real data, it is difficult to estimate the false positive rate based on a direct comparison. The number of pairs of breakpoints for which the statistic is computed is huge (≈ 40 million in the YRI ‘analysis panel’) while the number of predicted inversions is small (78 with a p -value of 0.02 or smaller). One cannot compare the ratio of the number of breakpoints examined to the predicted inversions in the HapMap and the simulated ‘analysis panels’ . Therefore, we use an indirect estimate.

For a p -value cut-off π , denote $\gamma(\pi)$ to be the ratio of the number of predicted regions with a p -value at most π in the HapMap ‘analysis panel’ to the corresponding number in the simulated data. If a lower p -value implies a greater chance of a prediction being real, one would expect $\gamma(\pi)$ to increase with decrease in π . Note that if the number of true predictions (which is unknown) is small or if the p -values for the real predictions are not concentrated in the tail of the distribution, it would be difficult to observe an increase in $\gamma(\pi)$. For the YRI ‘analysis panel’ , $\gamma(\pi)$ ranges from 1.73–1.75 for π in the range 0.1 – 0.06, but increases to $\gamma(0.02) = 2.85$, and $\gamma(0.01) = 4.86$. For a p -value of 0.02, this represents a 1.7-fold enrichment in the number of predictions in the HapMap data vs the simulated data. Under the assumption that the increase in the number of predictions in the tail of the p -value distribution is a result of true predictions, the false positive rate at cut-off of 0.02 can be estimated to be $\sim 58\%$. For the CEU ‘analysis panel’ , we didn’t observe a gradual increase in $\gamma(\pi)$ and also the number of predictions smaller than 0.02 is only 26, making it difficult to get a meaningful estimate of the false positive rate via this method. For the CHB+JPT ‘analysis panel’ , this method suggests a higher false positive rate of 80% at a cutoff of 0.02. This could reflect the low power of our method to detect true inversion polymorphisms in the CHB+JPT haplotype ‘analysis panels’ due to less accurate long range haplotype phasing in the CHB+JPT ‘analysis

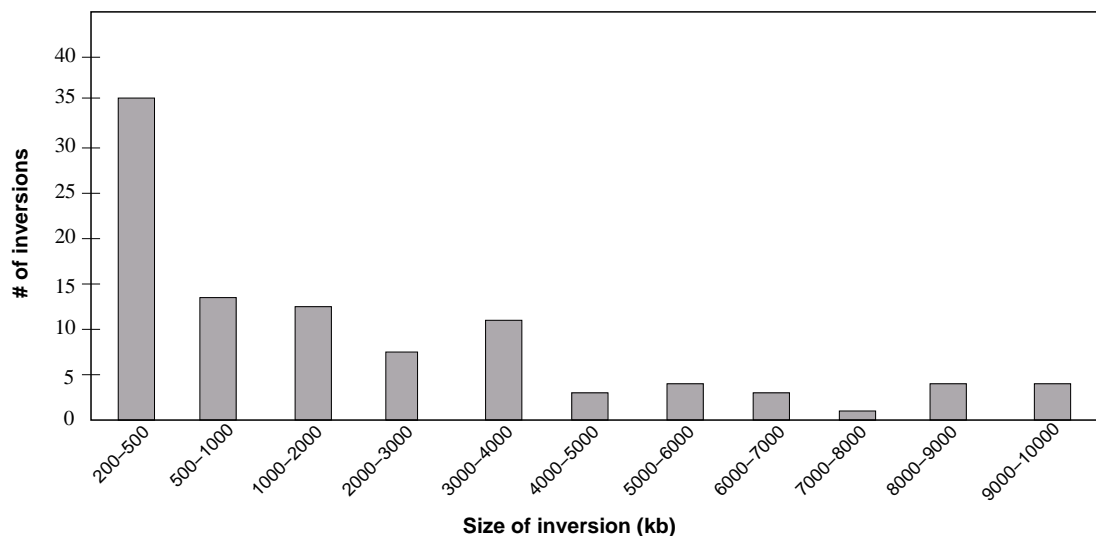


Figure 4.8: Length distribution of predicted inversions in the YRI ‘analysis panel’ . For this plot, we consider inversions with length in the range 200kb to 10Mb.

panels’ as compared to the CEU and YRI ‘analysis panels’ . Our analysis suggests that the false positive rate is the smallest in the YRI ‘analysis panel’ and about half of the YRI predicted inversions could be real. This is also supported by the fact that the two previously known inversions (that we detect across the 3 HapMap ‘analysis panels’) are detected in the YRI ‘analysis panel’ and about 10 predicted inversions in the YRI ‘analysis panel’ are supported by the presence of inverted repeats.

We also looked at the length distribution of the predicted inversions using our statistic in each of the three HapMap ‘analysis panels’ independently. For this we considered inversions with length in the range 200kb-10Mb. For the YRI ‘analysis panel’ , the number of predicted inversions seems to drop after 4Mb and remains essentially constant after that (see Figure 4.8). The number of predicted inversions with length in the range 1-4 Mb is 30 while the number of predicted inversions in the range 4-8 Mb is only 10. In contrast, for the CHB+JPT ‘analysis panel’ , the numbers are 62 (in the range 1-4 Mb) and 51 (in the range 4-8 Mb). These results indicate that there is a 3-fold clustering of predicted regions in the smaller range for the YRI ‘analysis panel’ . If most of the predictions were false, one would not expect to see any clustering. The higher

clustering in the YRI ‘analysis panel’ versus the CHB+JPT ‘analysis panel’ is consistent with the results from the coalescent simulations which also predict a smaller false positive rate for the YRI ‘analysis panel’ . While the above estimates of the false positive rate are crude, they nevertheless indicate that many of our predictions, especially those in the YRI ‘analysis panel’ , are likely to be real.

4.4 Discussion

We have presented a statistical method that has power to detect large inversion polymorphisms using population data. Our method can also detect large regions where the reference assembly has erroneous orientation. Applying our method to the HapMap data, we have identified 176 putative inversions in the three HapMap ‘analysis panels’ . The false positive rate for the predicted inversions in the YRI sample indicates that ≈ 30 of the 78 YRI predictions could represent real inversions. We have looked for independent evidence in the form of discordancies between the NCBI and Celera assembly, discordant fosmid pairs and presence of inverted repeats near inversion breakpoints for our predicted inversions. We have identified a novel 1.2 Mb long inversion on chromosome 10 that is supported by two discordant fosmid pairs and has not been reported before. For two of our predicted inversions, both breakpoints span gaps in the human reference assembly and the inverted orientation is represented in the Celera genome assembly, indicating orientation errors in the reference assembly. For about 10 regions, the inversions breakpoints are flanked by a pair of highly homologous inverted repeats. A recently proposed method called ‘haplotype fusion’ can assay single haplotypes for the presence of an inversion even when the breakpoints lie within long inverted repeats (Turner et al., 2006). The set of predicted inversions flanked by inverted repeats represent ideal candidates for validation using this technique.

Our method is designed to detect long inversions for which the inverted allele (w.r.t the reference sequence orientation) in a population has high frequency. Therefore, it is unlikely to detect inversion polymorphisms for which the inverted allele is the minor

variant. However, the allele frequencies of structural polymorphisms can vary significantly across populations. For 5 of the 10 deletion polymorphisms that were genotyped in the HapMap ‘analysis panels’, the minor allele in one ‘analysis panel’ was the major allele in another ‘analysis panel’ (McCarroll et al., 2006). The availability of data from multiple populations increases the chance of detecting the inversion using our method in the population where the inverted allele is the major variant. Furthermore, in many cases the reference sequence assembly is likely to represent the minor variant in the population. For a 18-kb inversion polymorphism at 7q11 (Feuk et al., 2005), the minor allele (frequency of 30%) was represented in the reference assembly while the major allele matches the orientation in the chimpanzee sequence. Although the method seems to be robust to the variation in recombination rates, it is possible that this heterogeneity in recombination rates and other events can produce a signal using our statistic. One such scenario is where the two breakpoints represent gene conversion hotspots while there is no recombination across the entire region. Gene conversion events would reduce short range LD while absence of recombination would maintain long range associations.

From a computational perspective, our method represents a novel strategy for using population data for detecting large rearrangements. It is becoming increasingly cost-effective to generate genome-wide SNP genotype data and our method can be applied to any such data. Other strategies have been suggested for computationally mining SNP data for potential inversions. Inversion polymorphisms have been extensively investigated for *Drosophila*, and it has been observed that the presence of inversion polymorphisms leads to strong and extended Linkage Disequilibrium across the inverted region since recombination in inversion heterozygotes is suppressed (Andolfatto et al., 2001; Navarro and Gazave, 2005; Navarro et al., 2000b). This reduces the overall recombination rate in the region and also tends to produce two divergent haplotype clades (Navarro et al., 1997; Andolfatto et al., 2001). The best known example of this effect in the human genome is the 900kb polymorphic inversion on chromosome 17 (Stefansson et al., 2005). However, it remains to be seen if this pattern is true of all (or most) human inversion polymorphisms. In fact, our analysis of haplotype patterns of the few known

inversion polymorphisms does not indicate that all inversion polymorphisms lead to such distinctive haplotype patterns (unpublished data).

Our results also indicate that many large inversion polymorphisms remain to be discovered in the human genome, and it may require extensive re-sequencing in multiple populations to find all such inversions. The presence of a large number of inversion polymorphisms could have major implications for evolution of the human genome. Inversions are known to directly suppress recombination in inversion heterozygotes. The lowering of recombination between inversion heterozygotes may also create effects similar to population sub-structure even without geographic isolation of the individuals. Characterization of inversion variants in human populations will be required to determine to what extent large inversions affect the recombination landscape of the human genome. Inversions could also represent an alternative mechanism for creating diversity in gene regulation, and splice isoforms. Such variation may also influence phenotypes and associations with diseases.

4.5 Appendix

4.5.1 Haplotype Data

We utilized genotype data from Phase I of the International HapMap project consisting of 269 individuals genotyped on about 1 million SNPs. These individuals consist of 30 trios from Utah region (CEU), 30 trios from Ibadan, Nigeria (YRI), 44 unrelated individuals from Tokyo, Japan (JPT) and 45 Han Chinese individuals from Beijing area (CHB). Since the JPT and CHB populations are genetically similar, we pooled the data from these two populations to obtain a larger ‘analysis panel’ of 89 individuals. For the CEU and YRI ‘analysis panels’, we used the 60 unrelated parents from the respective populations. We analyzed each of the three ‘analysis panels’: CEU, YRI and CHB+JPT separately. We used the phased haplotype data for these ‘analysis panels’ (HapMap data release #16 available at http://www.hapmap.org/downloads/phasing/2005-03_phaseI/full/).

Since the SNPs in this data were ordered based on the NCBI Build 34 (hg16) assembly of the human genome, all our results are with respect to NCBI Build 34 assembly. We used the phased data since it is difficult to detect long range LD without phasing information. The phasing is highly accurate for the CEU and the YRI ‘analysis panels’ due to the presence of trio information. For the JPT and CHB populations, in the absence of trios, the haplotype phasing is less accurate (a switch error every 0.34 Mb (The International HapMap Consortium, 2005)). This can destroy long range LD, thereby potentially reducing the power of our method to detect inversions in the CHB+JPT ‘analysis panel’ .

4.5.2 Identifying potential inversions

For every chromosome, we considered the region between every pair of adjacent SNPs as a potential breakpoint. If a pair of adjacent SNPs showed high correlation using the r^2 measure (a cutoff of $r = 0.6$ was used), the region in between is highly unlikely to be a breakpoint and was excluded. For every breakpoint, we choose a multi-SNP marker to the left of the breakpoint and another one to the right of the breakpoint (these were chosen to be the physically closest multi-SNP markers to the breakpoint from the set of multi-SNP markers defined previously). Each breakpoint is reported as a pair of genomic coordinates corresponding to the right physical boundary of the multi-SNP marker closest to the left of the breakpoint and the left physical boundary of the multi-SNP marker closest to the right of the breakpoint. For every pair of breakpoints within a certain maximum distance, we computed the two log-likelihood ratios and the corresponding p -value. All pairs of breakpoints with low p -value are considered as potential candidates for inverted regions. A predicted inversion is reported as a 4-tuple (l_1, l_2, r_1, r_2) corresponding to a pair of left (l_1, l_2) , and right (r_1, r_2) breakpoints.

For analysis of the HapMap data, we ignored pairs of breakpoints within 200kb of each other since considerable LD is observed at short distances in the HapMap data and power simulations also indicate that our method has low power to detect inversions

of small length. Our results for the estimation of the false positive rate indicated that there was some enrichment for true positives in the predicted inversions with p -value smaller than 0.02. Therefore we choose a p -value cutoff of 0.02 for generating the predicted inversions. We also limit the size of the largest predicted inversion that we consider to 4 megabases. The largest known polymorphic inversion in the human genome is about 4.5Mb in length (Genome Variation Database at <http://projects.tcag.ca/variation/>). Also, the distribution of the length of the predicted inversions suggests that predicted inversions larger than 4Mb represent false positives rather than true inversions. All pairs of predicted inversion breakpoints with length in the range 200kb-4Mb and with a p -value of 0.02 or smaller were enumerated for each chromosome in the three HapMap ‘analysis panels’. For each ‘analysis panel’ and chromosome, we clustered the predicted inversions based on the physical location of the breakpoints. For two predicted inversions (l_1, l_2, r_1, r_2) and (p_1, p_2, q_1, q_2) , if the segment (l_1, l_2) and (p_1, p_2) overlapped and similarly if (r_1, r_2) and (q_1, q_2) overlapped, these two predicted inversions were grouped together. After clustering, we had 215 predicted inversions in the three ‘analysis panels’. For every cluster we report the pair of inversion breakpoints with the smallest p -value. In order to further reduce potential false positives, we removed predicted inversions for which there was strong LD between the block to the left of the left breakpoint (block 1 in Figure 4.2) and the block to the right of the right breakpoint (p -value of the multi-allelic LD smaller than 0.02).

4.5.3 Sequence Analysis

We used the repeat-masked June 2003 (NCBI Build 34) human genome sequence from the UCSC (University of California, Santa Cruz) Human Genome Browser website for analyzing the inversion breakpoints. For each predicted inversion, the genomic sequence in the window $[l_1 - 200000 \dots l_2 + 200000]$ was blasted against the sequence in the window $[r_1 - 200000 \dots r_2 + 200000]$ to find pairs of homologous sequences. Only hits with an e-value less than $1e-25$ and length at least 100bp were

considered. We also removed pairs of homologous sequences that were less than 100kb apart. The statistical significance of the number of inversion breakpoints flanked by a pair of inverted repeats was estimated empirically as follows. We simulated 1000 random lists of inversions and computed the number of inversions with a pair of inverted repeats. Each random list of inversions was generated by shifting each predicted inversion (on the HapMap ‘analysis panels’) to a random location on the same chromosome on which it was detected. The p -value was estimated to be 0.006 using this method. Additionally, we observed that the length of the inverted repeats for many of the predicted inversions was generally much longer than those for the random lists.

Analysis of genes near inversion breakpoints was performed using the UCSC KnownGenes II list from the UCSC Genome Browser. A gene was defined to cover an inversion breakpoint, if the transcriptional start position of the gene was before the left boundary of the breakpoint and the transcriptional end location after the right boundary of the breakpoint. In order to assess the statistical significance of the number of inversion breakpoints covered by one or more genes, we used an empirical method similar to the one used above for inverted repeats. We simulated 1000 random lists of inversions and computed the number of genes covering breakpoints for each list.

4.5.4 Coalescent Simulations

We simulated population data using the Cofi program (Schaffner et al., 2005) which implements a coalescent model similar to the MS program (Hudson, 1990) but allowing for complex demographic histories and variable recombination rates. We used the bestfit model which has been calibrated using genome-wide human population data for different populations. The bestfit model uses the large-scale variation in recombination rates obtained from the deCODE genetic map along with fine-scale variation in recombination rates. We used the default parameters of this model which are listed in Table 1 of the paper describing the method (Schaffner et al., 2005). The program generates data for four populations, each with its own demographic scenario. We used the

data for three of the populations: West African, European and East Asian. These three populations were considered as proxies for the YRI, CEU and CHB+JPT ‘analysis panels’ respectively from the International HapMap project. We matched each HapMap ‘analysis panel’ to the corresponding simulated ‘analysis panel’ in the number of chromosomes. We didn’t model SNP ascertainment bias (present in the HapMap ‘analysis panels’) for the simulated data since it is unlikely to affect our results as we discard SNPs with low minor allele frequency (less than 0.1). We generated 100 datasets of length 20Mb (it is computationally infeasible to generate chromosomal length regions using the *cosi* program) for each of the three ‘analysis panels’ . We simulated data with a fixed number of SNPs and then thinned the SNPs so that the average SNP density (for SNPs with minor allele frequency ≥ 0.1) matched that of the HapMap data.

4.6 Acknowledgement

Chapter 4, in full, was published in *Genome Research*, Vol 17(2), pp 219-30, 2007, V. Bansal, A. Bashir and V. Bafna, “Evidence for large inversion polymorphisms in the human genome from HapMap data”. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Haplotype Assembly from Whole-genome sequence data

5.1 Introduction

Humans are diploid organisms with two copies of each chromosome (except the sex chromosomes). In recent years, the development of high-throughput technologies has made it incredibly easy and cost-effective to read the DNA sequence at millions of SNPs across the genome. However, these genotyping methods determine the two alleles at a individual SNP and are unable to provide information about haplotypes, the combination of alleles present at multiple SNPs along a single chromosome. Haplotypes observed in human populations are a result of shuffling of ancestral haplotypes through recombination and contain much more information about human genetic variation than genotypes. In chapters 2 and 3, we have seen how population haplotypes are useful for detecting historical recombination events and identifying recombination hotspots. In chapter 4, we also saw how highly accurate and long genome-wide haplotypes are useful for searching for large inversion polymorphisms in the genome. Haplotypes from the HapMap project have proven to be invaluable for whole-genome association studies in multiple ways. To reduce cost, disease association studies are performed using a sub-

set of SNPs in the human genome. The HapMap haplotypes are useful for evaluating the power of these subsets to detect association at the untyped SNPs in human populations. Further, the haplotype data has also been used for fine-scale mapping of variants identified in association studies (Gudmundsson et al., 2007) and improving the power of whole-genome association studies (Zaitlen et al., 2007; Pe'er et al., 2006; Marchini et al., 2007).

In the absence of molecular methods for determining haplotypes, haplotypes are inferred computationally from SNPs genotyped in a set of individuals from a population (Clark, 1990; Excoffier and Slatkin, 1995; Stephens et al., 2001; Niu et al., 2002; Stephens and Donnelly, 2003). All haplotype phasing methods, explicitly or implicitly, exploit Linkage Disequilibrium (LD), the correlation of alleles at physically proximal SNPs in the human genome. In short regions of the genome, high LD reduces the number of distinct haplotypes, allowing these methods to piece together haplotypes for an individual. The great variation in recombination rates and Linkage Disequilibrium across the human genome limits the accuracy of these methods. A popular haplotype phasing method, PHASE(Stephens et al., 2001), has a switch error rate of 5.4% for unrelated individuals from a European population(Marchini et al., 2006); this corresponds to one switch error between the maternal and paternal chromosomes approximately every 50kb. In general, population data from unrelated individuals does not contain enough information to reliably estimate the haplotypic phase between distant markers (>100kb). Accurate long-range haplotypes may prove useful for finding multiple genetic variants that contribute to complex diseases. For accurate long range haplotyping, additional information such as family data is invaluable. For example, the presence of trios in two of the HapMap populations (CEU and YRI) has allowed the inference of highly accurate haplotypes. However, family data is hard to obtain for every population sample.

The availability of full diploid genome sequences for a large number of individuals would be ideal for obtaining a comprehensive understanding of all forms of genetic variation and especially useful for finding rare genetic variants associated with disease. Advancements in sequencing technology are driving down the cost of sequenc-

ing and it should be possible to completely sequence many human individuals in a few years (Schuster, 2008; Shaffer, 2007). Whole-genome sequence data from a single individual represents an alternate resource from which the two haplotypes can potentially be determined. Each sequence read represents a fragment of a chromosome. A read that spans multiple variant sites can reveal the combination of alleles present at those sites on that chromosome. Using the overlaps at heterozygous sites between a collection of reads, one can potentially assemble the two haplotypes for a chromosome (see Figure 5.1 for an illustration). This “haplotype assembly” represents a different computational challenge in comparison to genome sequence assembly, where one uses the sequence overlap between reads (ignoring the variant sites) to piece together a haploid genomic sequence.

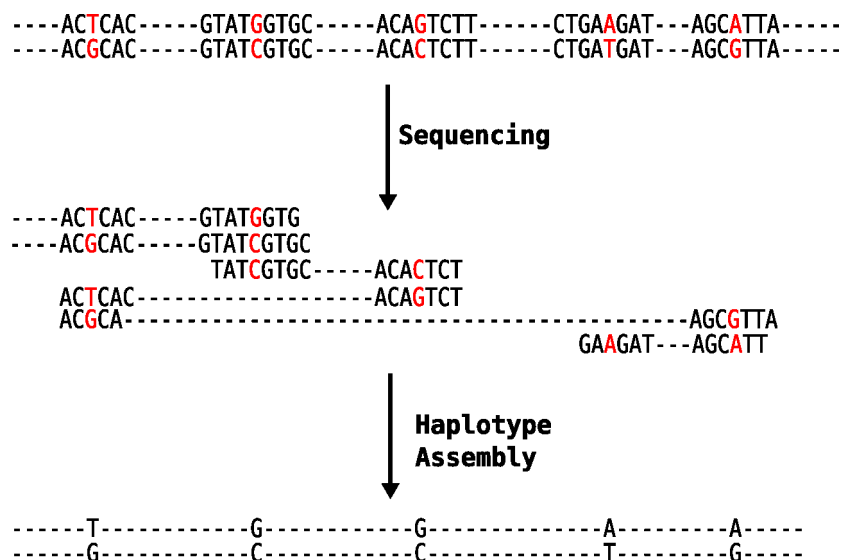


Figure 5.1: Illustration of how haplotypes can be assembled from sequenced reads. Each read is a fragment of one of the two chromosomes. Reads that share a allele at a common variant can be inferred to come from the same chromosome and joined together. Reads that differ at a particular variant can be inferred to come from different chromosomes and similarly extend the two haplotypes.

Haplotype assembly refers to the problem of reconstructing haplotypes from a collection of sequenced reads given a genome sequence assembly. A more challenging problem is to separate out the two haplotypes during the sequence assembly process it-

self. This has recently been done for some small, highly polymorphic genomes (Vinson et al., 2005), but remains difficult to accomplish for large eukaryotic genomes such as humans. Large eukaryotic genomes include many repetitive sequences, and a sequence assembly must therefore distinguish between two (almost identical) instances of a sequence that lie on the same chromosome as well as separating the chromosomes. The haplotype assembly problem may seem easier but the objectives are different. By working with a reference sequence, one can focus on obtaining highly accurate haplotypes and estimating their reliability rather than just obtaining ‘a single’ haplotype assembly. Also, as many individuals in a population are sequenced, it is computationally more efficient to generate a reference assembly once, and assemble haplotypes for each of the individuals.

For haplotype assembly to be feasible, one requires a high sequence coverage (sufficient overlaps between reads) and reads that are long enough to span multiple variant sites. Given the level of polymorphism in the human genome ($\sim 0.1\%$), single shotgun reads (about 800-1000 base pairs long) at 5-8x coverage would result in short haplotype segments. However, *paired ends* or mate pairs (pair of sequenced reads derived from the same shotgun clone) provide linkage information that can substantially increase the length of inferred haplotypes. Even with mate pairs, it is not possible to link all variants on a chromosome. A haplotype assembly for a diploid genome is a collection of haplotype segments or disjoint haplotypes. In the absence of errors in sequenced reads, the correct haplotype assembly is unique and is not difficult to derive. Errors in reads increase the space of possible solutions making this problem computationally challenging. The problem of finding the haplotype assembly that optimizes a certain objective function (e.g. minimize the number of conflicts with the sequenced reads) has been explored from a theoretical perspective (Bafna et al., 2005; Rizzi et al., 2002; Halldorsson et al., 2003; Lippert et al., 2002), and has been shown to be computationally intractable for gapped reads (e.g. mate pairs). A statistical method was proposed (Li et al., 2004) for reconstructing haplotypes from sequenced reads aligned to a reference genome. The method is based on inferring local haplotypes using a Gibbs

sampling approach and joining these local haplotypes using overlaps. This method has recently been extended (Kim et al., 2007) to include polymorphism detection as part of the haplotype reconstruction pipeline and applied to the genome of *Ciona intestinalis*.

Recently, Levy and colleagues (Levy et al., 2007) have sequenced the complete diploid genome of a single human individual. Approximately 32 million sequenced reads (from clone libraries of various lengths), were used to generate a genome assembly referred to as “HuRef”. More than 4.1 million genomic variants were detected by identifying heterozygous alleles within the sequenced reads and through comparison of the HuRef assembly with the NCBI version 36 human genome assembly. Of these, 1.8 million heterozygous variants were used for haplotype assembly. The presence of paired-end sequences or mate-pairs with different insert sizes (ranging from 2kb to 40kb) increases the length of the haplotype segments that can be inferred, but also results in links between physically distant variants. As mentioned earlier, there are no efficient algorithms for haplotype assembly in the presence of mate-pairs, and statistical methods for haplotype assembly (Li et al., 2004; Kim et al., 2007) which start by inferring short local haplotypes are not particularly suited for the HuRef data. A simple greedy heuristic was implemented to build haplotypes incrementally starting from single reads (see Material and Methods, Levy et al., 2007 Levy et al. (2007)). More than 70% of the 1.8 million heterozygous variants used for haplotype assembly were assembled into haplotypes that cover at least 200 variants. In addition, 1.5 Gb of the genome could be covered by haplotypes longer than 200kb in length. Comparison of sequenced reads to the reconstructed haplotypes showed that 97.4% of the variant calls are consistent with the haplotype assembly. Notwithstanding the reasonable accuracy of the haplotype assembly for HuRef, the greedy strategy represents a relatively simple approach for this problem. It incrementally reconstructs a single haplotype assembly and does not attempt to find a haplotype assembly that is optimal under a probabilistic or combinatorial model. In Levy et al., 2007 Levy et al. (2007), we had briefly mentioned that it is possible to obtain a more accurate haplotype assembly using Markov chain Monte Carlo (MCMC) methods and had implemented one such algorithm. In this chapter, we

describe a novel MCMC algorithm, *HASH (Haplotype Assembly of Single Human)* for haplotype assembly. The MCMC approach represents a natural way to search the space of possible haplotypes to find likely haplotype reconstruction(s) and also allows us to estimate the reliability of the reconstructed haplotypes. The transitions of the Markov chain underlying our algorithm are determined using the graph structure of the links between the variants and are not restricted to be local.

Results on the HuRef sequence data demonstrate that the haplotypes reconstructed using HASH are more consistent with the sequenced fragments than the haplotypes obtained using the greedy heuristic. Using haplotypes sampled by the MCMC algorithm, we estimate that the HuRef haplotypes have a switch error rate of 0.9%. Using simulations, we also demonstrate that our MCMC algorithm can reconstruct haplotypes to a high degree of accuracy and determine which variant calls are likely to be incorrect. Based on comparison to population haplotypes from the HapMap project, we estimate a switch error rate of approximately 1.1% for the HuRef haplotypes inferred using HASH. In comparison, the switch error rate for the haplotypes reconstructed using the greedy heuristic is 3.1%. Although we describe results using data from a human genomic sequence, our methods are valid for performing haplotype assembly from sequenced reads generated using any sequencing technology as long as the polymorphism rate for the sequenced organism and the length of sequenced reads allow the linking of multiple variants. They are also applicable to inferring haplotypes using short haploid sequences from other sources (see e.g. Konfortov et al., 2007 Konfortov et al. (2007)).

5.2 Methods

We assume that a list of genetic variants such as SNPs, short insertions/deletions, etc is available. A list of polymorphic variants can be generated while performing sequence assembly or can be obtained from a database of genetic variants such as dbSNP (Sherry et al., 2001). We restrict ourselves to variants that have been identified to be heterozygous in the genome of the individual under consideration as homozygous

variants are uninformative about phasing of other variants. Note that certain variants that are truly heterozygous in the genome may be reported as homozygous as both alleles are not sampled sufficient number of times during sequencing.

Each sequenced read is mapped to the reference genomic sequence to obtain the alleles it has at each of the heterozygous sites. For a variant, reads with sequence matching the consensus sequence are assigned as '0' while those not matching are assigned as '1'. The assumption of just two alleles makes sense in the absence of errors. However, the presence of base-calling errors makes actual data more complex with tri-allelic variants. Here, we assume that such sites are filtered out (described in Levy et al. (Levy et al., 2007). Paired-end reads from the same clone that map to the assembly in the expected orientation and whose physical separation is within the expected range are represented as a single fragment. Mated reads that show some inconsistency in orientation or distance are split into two separate fragments. Note that these aberrant mapping pairs might represent chimeric errors, but also, heterozygous structural variation in the HuRef genome; Levy et al., 2007Levy et al. (2007) describe some of these variations. Here, we ignore this additional information.

5.2.1 Haplotype Likelihood

Formally, each fragment i is represented by a ternary string $X_i \in \{0, 1, -\}^n$, where the $-$ corresponds to the heterozygous loci not covered by the fragment. The complete data can be represented by a *fragment matrix* X with m rows and n columns where each row represents a fragment and each column corresponds to a variant site. Corresponding to each variant call $X_i[j]$, we have an error probability $q_i[j]$, which denotes the probability that the variant call is incorrect. As $q_i[j]$ cannot be estimated from the fragment data, we use quality scores $s_i[j]$ that usually accompany sequence data. For example, the quality scores might be obtained using Phred (Ewing and Green, 1998). Sequence quality scores are integer values related to the error probabilities as

$$q_i[j] = 10^{-\frac{s_i[j]}{10}}$$

For SNPs, $s_i[j]$ describes the quality value for the allele call; for multi-base variants, $s_i[j]$ is the lowest of the quality values for the base-calls in the variant; for the case of a gap (insertion/deletion), $s_i[j]$ corresponds to the lower of the two quality values on either side of the gap. If information about the sequencing quality values is not available or for performing simulations, we assume a uniform error probability $q_i[j] = \hat{q}$ for all variant calls. In what follows, we will assume that q is available and fixed.

Let $H = (h, \bar{h})$ represent the unordered pair of haplotypes where h is a binary string of length n and \bar{h} is the bit-wise complement of h . The problem of reconstructing the most likely pair of haplotypes given the fragment data (known) is given by

$$\arg \max_H Pr(X|H, q)$$

However, we are interested in sampling H from a probability distribution. Using Bayes rule, we can write

$$Pr(H|X, q) = \frac{Pr(X|q, H)Pr(H|q)}{\sum_{H'} Pr(X|q, H')Pr(H'|q)} \quad (5.1)$$

Assuming a uniform prior on the space of haplotypes, we have

$$Pr(H|X, q) \propto Pr(X|H, q) \quad (5.2)$$

In the following discussion, we will refer to $Pr(X|H, q)$ as a distribution over H for notational convenience. Define the function $\delta(X_i[j], h[j]) = 1$ if $X_i[j] = h[j]$ and 0 otherwise.

Then,

$$Pr(X_i|q, h) = \prod_{\{j: X_i[j] \neq -\}} \delta(X_i[j], h[j])(1 - q_i[j]) + (1 - \delta(X_i[j], h[j]))q_i[j] \quad (5.3)$$

Extending this to a haplotype pair $H = (h, \bar{h})$, we define

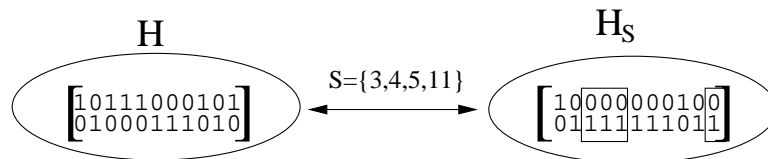
$$Pr(X_i|q, H) = \frac{(Pr(X_i|q, h) + Pr(X_i|q, \bar{h}))}{2} \quad (5.4)$$

Then, $Pr(X|q, H)$ can be computed as a product over fragments (assuming that fragments are independently generated):

$$Pr(X|q, H) = \prod_i Pr(X_i|q, H) \quad (5.5)$$

5.2.2 Markov chain Monte Carlo algorithm

Instead of computing the most likely solution, it is potentially more useful to sample from the posterior distribution of haplotypes. As the number of possible haplotypes grows exponentially with the number of variants, we construct a Markov chain to sample from the posterior distribution of H given the fragment matrix X and the matrix of error probabilities q . The states of the Markov chain correspond to the set of possible haplotypes. Transitions of the Markov chain are governed by subsets S of columns of the fragment matrix X . Specifically, each transition is of the form: $H \rightarrow H_S$ where H is the current state (haplotype pair) and H_S is a new haplotype pair created by ‘flipping’ the values of the columns in S . For example,



Note that at columns not in S , such as column 1, H and H_S are identical. However, columns in $S = \{3, 4, 5, 11\}$ are flipped in H_S .

If $\Gamma = \{S_1, S_2 \dots S_k\}$ is a collection of subsets of columns of X , then for each state H , there are $k + 1$ possible moves to choose from, including the self-loop. The Markov chain in state H chooses a subset $S_i \in \Gamma$ and moves to the new state H_{S_i} with a certain probability. The transition probabilities are chosen to ensure that they satisfy the detailed balance conditions. The MCMC algorithm is described as follows:

Initialization: Choose an initial haplotype configuration $H^{(0)}$.

Iteration : For $t = 1, 2, \dots$ obtain H^{t+1} from H^t as follows:

1. With probability $1/2$, set $H^{t+1} = H^t$
2. Otherwise sample a subset S from Γ with probability $\frac{1}{|\Gamma|}$
3. With probability $\min \left[1, \frac{Pr(X|H_S^t, q)}{Pr(X|H^t, q)} \right]$, set $H^{t+1} = H_S^t$. Otherwise set $H^{t+1} = H^t$

As can be seen, our algorithm uses the Metropolis update rule (Metropolis et al., 1953), and is completely specified by Γ , fragment matrix X and the matrix q of error probabilities. We denote the corresponding Markov chain as $\mathcal{M}(X, q, \Gamma)$ or simply by $\mathcal{M}(\Gamma)$, whenever X and q are implicit. Note that Step 1 of the above algorithm which represents a self-loop probability of $1/2$ is added to ensure aperiodicity which is required for analysis of the mixing time of the Markov chain (Randall, 2006). In practice, it is not essential and can be removed as most Markov chains are indeed aperiodic.

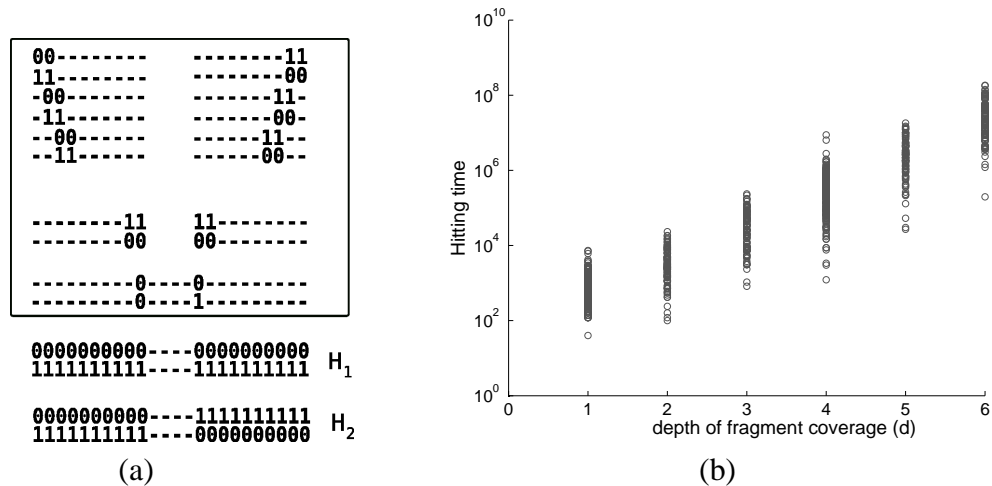


Figure 5.2: a) Example of a fragment matrix $X(n, d)$ for which two haplotypes H_1 and H_2 have equal likelihood. The matrix has n columns with each pair of adjacent columns (except the pair $(n/2, n/2 + 1)$) covered by d fragments ($d = 2$ and $n = 20$ as shown). b) Plot of hitting time (number of steps taken by the Markov chain $\mathcal{M}(\Gamma_1)$ to go from H_1 to H_2) as a function of the depth of fragment coverage (d). For each d value, the Markov chain was run 100 times (each value represented as a circle) with $\hat{q} = 0.05$.

5.2.3 Choosing Γ

A simple choice for Γ is $\Gamma_1 = \{\{1\}, \{2\}, \dots, \{n\}\}$. It is easy to show that any Markov chain $\mathcal{M}(X, q, \Gamma)$ is ergodic and has the desired posterior distribution $Pr(H|X, q)$ if $\Gamma_1 \subseteq \Gamma$ (See Supplementary Methods for proof). However, we also want the Markov chain to have a low “mixing time”, i.e after a small number of steps, the Markov chain should be “almost” sampling from the stationary distribution. Empirical results suggest that the chain $\mathcal{M}(\Gamma_1)$ does not sample the haplotype space quickly enough and in particular takes a large number of steps to move from a haplotype H to another haplotype H' such that H and H' differ in a large block of columns and $Pr(H'|X, q) \approx Pr(H|X, q)$. The reason for this is that in order to sample H' , the Markov chain has to go through several intermediate haplotype configurations, each of low-probability. As an illustrative example, Figure 7.1(a) describes the case of a fragment matrix for which there are two equally likely haplotype configurations H_1 and H_2 which differ by a single flip of half of the columns. The fragment matrix which we denote by $X(n, d)$ has n columns and every column is spanned by d fragments ($n = 20$ and $d = 2$ in the example shown).

For this fragment matrix, we empirically estimate the hitting time of the Markov chain to reach H_2 starting from H_1 as a function of d (see Figure 7.1(b)). The error probability \hat{q} is identical for all positions. As d increases, the likelihood of all haplotypes other than the two haplotypes H_1 and H_2 decreases. Therefore, $\mathcal{M}(\Gamma_1)$ will take increasingly longer time to move from H_1 to H_2 . The plot shows that the hitting time increases exponentially with d . For $d = 5$, the expected hitting time (averaged over 100 runs) is roughly 10 million steps. For larger sized examples and smaller q , the hitting time will be much greater. The above experiments suggest that the mixing time of $\mathcal{M}(\Gamma_1)$ is unlikely to be small. Indeed, we prove theoretically that the mixing time of $\mathcal{M}(\Gamma_1)$ grows exponentially with d (see Supplementary Methods for formal definition of mixing time and proof of this result).

Clearly, the Markov chain $\mathcal{M}(\Gamma_1)$ is not a desirable chain for sampling the hap-

lotype space. A natural question to ask is: can the mixing time be reduced with inclusion of larger sized subsets in Γ ? For the fragment matrix $X(n, d)$ shown in Figure 7.1, if we include the subset $S_{1\dots n/2}$ (columns 1 to $n/2$) in Γ , the Markov chain will move from H_1 to H_2 whenever this subset is sampled. Therefore, the hitting time should reduce to $O(n \log n)$, which we empirically observe to be true (results not shown). Intuition suggests that the mixing time of the Markov chain should also decrease since the bottleneck in moving between H_1 and H_2 has been removed. Using a combination of methods for analyzing the mixing time of Markov chains, it can be shown that the mixing time of the Markov chain $\mathcal{M}(\Gamma_1 \cup S_{1\dots n/2})$ is indeed polynomial in n and d . (V. Bansal and V. Bafna, unpublished results).

These results demonstrate that $\mathcal{M}(\Gamma_1)$ has poor mixing time for a specific fragment matrix, but augmenting Γ_1 slightly can dramatically improve the mixing time. In addition, we gain an insight into how Γ_1 should be augmented. Results on mixing time of Markov chains (Sinclair and Jerrum, 1989; Sinclair, 1992) imply that if there is a subset of haplotype(s) \mathcal{H} for which the probability of moving to a haplotype outside \mathcal{H} in a single step (scaled by the probability of being in state \mathcal{H}) is low, the Markov chain has poor mixing time. For the fragment matrix $X(n, d)$ (Figure 7.1) and the Markov chain $\mathcal{M}(\Gamma_1)$, H_1 represents such a subset \mathcal{H} . While there are such “bottlenecks” in the transition matrix, the mixing time is bad. Subsets of columns (such as $S_{1\dots n/2}$) that remove such bottlenecks represent good candidates to include in Γ . However, for a general fragment matrix it is not obvious how to choose these subsets computationally and how many subsets to include. We do not want the number of subsets in Γ to be very large since the probability that a specific subset is picked decreases with increasing $|\Gamma|$, thereby increasing the mixing time. Therefore, we restrict ourselves to choosing Γ with size linear in the number of columns of the fragment matrix.

To motivate our approach for constructing Γ , we return back to the example of the fragment matrix $X(n, d)$ in Figure 7.1. As we saw before, $S_{1\dots n/2}$ represents a good candidate to include in Γ since it allows the Markov chain to move easily between the two haplotype configurations H_1 and H_2 . The set of columns in this subset is linked

to the rest of the columns by two fragments and in addition, these two fragments are inconsistent with each other. Subsets that contain columns from both $S_{1\dots n/2}$ and from outside this subset are not particularly important for adding to Γ . Therefore, the problem of constructing Γ can be considered independently for the fragment matrix restricted to $\mathcal{S} = S_{1\dots n/2}$ and the matrix restricted to $\bar{\mathcal{S}}$. This suggests a recursive partitioning strategy for constructing Γ , which we describe next.

5.2.4 A graph-partitioning approach

We construct an undirected weighted graph $G(X)$ with each column of the fragment matrix as a separate node of this graph and an edge between two nodes if there is some fragment that covers both columns. The weight of an edge between two columns is the number of fragments that cover both columns. A *cut* in $G(X)$ is simply a subset S of vertices, with weight equal to the sum of weights of the edges going across the cut. A *Minimum-cut* is a cut with minimum weight in the graph $G(X)$. From the perspective of the Markov chain, a cut represents a subset of variants, and a cut with low-weight represents a good candidate to include in Γ . We partition the graph $G(X)$ into two pieces S and \bar{S} using a simple min-cut algorithm (Stoer and Wagner, 1994) and add the two subsets S, \bar{S} to Γ . We apply the same procedure recursively to the two induced sub-graphs $G(S)$ and $G(\bar{S})$ adding two new subsets to Γ every time we compute a new cut. The recursive graph-partitioning approach ensures that Γ includes Γ_1 and has n additional subsets.

We construct a graph $G(X)$ with vertex set V as the set of all columns of X and edge set E as all pairs of columns (i, j) of X for which there is at least one row in X covering both columns, i.e. $X_k[i] \neq -$ and $X_k[j] \neq -$ for some row k . The weight of the edge (i, j) is the number of such rows covering both columns. A *cut* (S, \bar{S}) in $G(X)$ is a partition of the vertices of $G(X)$ into two disjoint set of vertices. The weight of a cut (S, \bar{S}) is equal to the sum of the weights of edges going across the cut from S to the \bar{S} .

GraphPartitioning(X, Γ)

1. If the number of columns in X is less than 2, return Γ
2. Compute a min-cut (S, \bar{S}) in the graph $G(X)$
3. $\Gamma = \Gamma \cup \{S, \bar{S}\}$
4. GraphPartitioning($X(S), \Gamma$)
5. GraphPartitioning($X(\bar{S}), \Gamma$)
6. return Γ

Information about the variant calls in the fragment matrix can be used for assigning weights to the edges in $G(X)$. This is potentially more informative than just using the number of fragments. Consider the example fragment matrix in Figure 7.1. The subset $S_{1\dots n/2}$ is a good candidate for Γ not only because the cut corresponding to this subset has low weight (two edges) but also because the two fragments linking this subset of columns to the rest of the matrix are inconsistent with each other. We have developed a scheme that assigns weights to the edges of $G(X)$ based on the consistency of a haplotype pair H with the fragment matrix. A fragment adds 1 to the edge weight between two columns if the phase suggested by the fragment is consistent with the current haplotype assembly. If not, it contributes -1 to the edge weight. Hence, a cut with low or negative weight corresponds to a subset of columns whose current phase with respect to the rest of the columns is inconsistent with the fragment matrix.

Consider a pair of columns i, j in X . W.l.o.g, assume that the current haplotype H is described by $H = [00\dots 0, 11\dots 1]$. For a fragment f , we describe a function *match* that scores f on being consistent (or not) with the haplotype at positions i, j .

Formally

$$\text{match}_H(f, i, j) = \begin{cases} 1 & \text{if } (f[i], f[j]) \in \{00, 11\} \\ -1 & \text{otherwise} \end{cases}$$

We use the match function for assigning weights to edges in the graph $G(X)$. Define

$$w_H(i, j) = \sum_f \text{match}_H(f, i, j)$$

If $w_H(i, j)$ is highly positive, it implies that the current haplotype pair H is consistent with the phasing suggested by the fragments for the pair (i, j) . On the other hand, a negative value for $w_H(i, j)$ indicates that the phasing for the pair (i, j) in the haplotype pair H is likely to be incorrect. We denote the graph with edge weights $w_H(i, j)$ by $G(X, H)$, and would like to compute Min-Cuts. As negative weights on $w_H(i, j)$ make the problem equivalent to the Max-Cut problem, which is known to be computationally hard (Garey and Johnson, 1979), we use the heuristic of removing all edges (i, j) for which $w_H(i, j) < 0$ from $G(X, H)$. We denote the graph partitioning algorithm based on $G(X, H)$ for computing Γ as *WeightedGraphPartitioning*(X, H).

The recursive graph-partitioning approach for constructing Γ is greatly motivated by the nature of the sequencing data that we have analyzed. The example that we presented in Figure 7.1 is quite typical of real data. Figure 5.4 depicts an example of a fragment matrix from chromosome 22 of HuRef. Shotgun sequencing leads to non-uniform sampling of variants creating “weak” links in the fragment matrix that the graph-partitioning approach can exploit to construct Γ .

5.2.5 The complete MCMC algorithm

The collection of subsets Γ computed using the weighted graph-partitioning approach is dependent upon the haplotype pair H . As we sample haplotypes with greater likelihood, it is potentially useful to update Γ . The complete algorithm which we call “HASH” (short for **H**aplotype **A**ssembly for **S**ingle **H**uman) is as follows:

HASH(\mathbf{X}, \mathbf{q})

1. Set $\Gamma^{(0)} \leftarrow \Gamma_1$.
2. Set $H^{(0)}$ at random or otherwise.

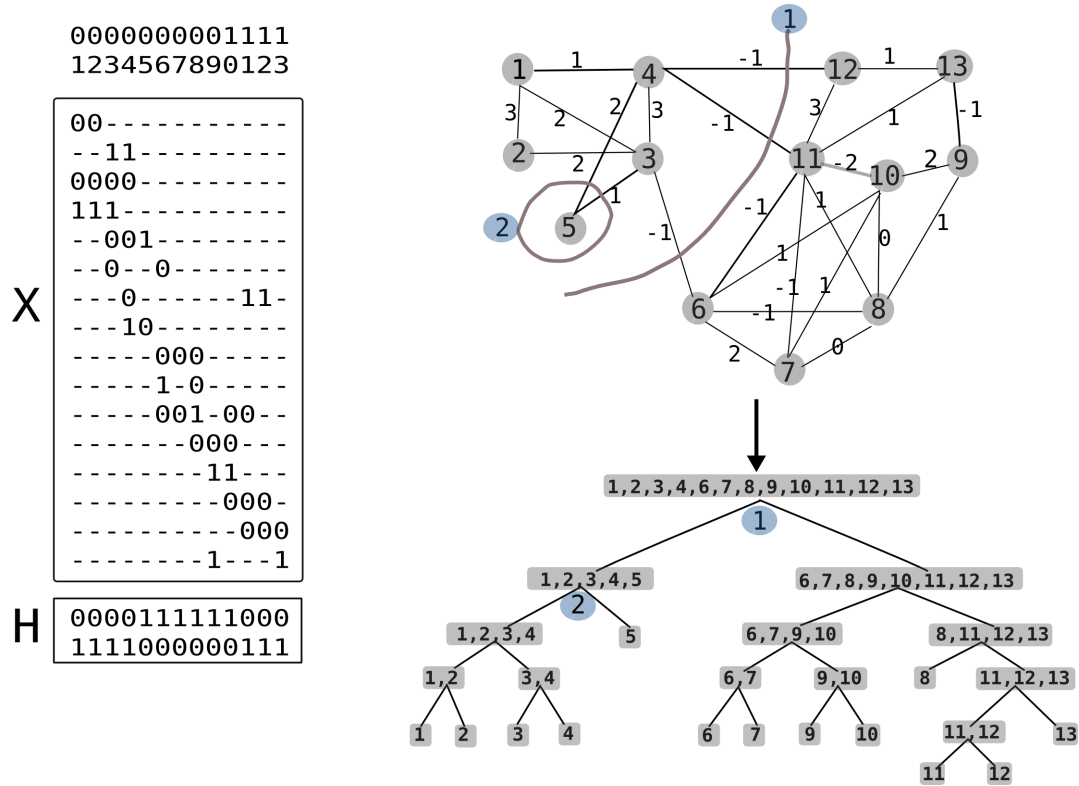


Figure 5.3: Illustration of the weighted graph $G(X)$ derived from the fragment matrix X and a haplotype pair H and the recursive graph-partitioning algorithm for computing Γ . The graph is shown on the top right and the tree structure below demonstrates the recursive partitioning of the graph using min-cut computations. The first cut (labeled as 1), partitions the columns of X into two subsets: $S = \{1, 2, 3, 4, 5\}$ and $\bar{S} = \{6, 7, 8, 9, 10, 11, 12, 13\}$. The second cut (labeled 2), further partitions the subset S into two smaller subsets: $\{1, 2, 3, 4\}$ and $\{5\}$. Γ is obtained from the subsets labeling the nodes of the tree (except the root node).

3. For $t = 1, 2, \dots$
 - (a) Let $H^{(t)} = \mathcal{M}(\Gamma^{(t-1)}, X, H^{(t-1)}, c)$ be the haplotype obtained after running $\mathcal{M}(\Gamma^{(t-1)})$ for $c \cdot n$ steps ($c \approx 1000$).
 - (b) Compute $\Gamma^{(t)} = \text{WeightedGraphPartitioning}(X, H^{(t)})$.
4. Set $\Gamma \leftarrow \Gamma^{(t)}$ and discard all previous samples.
5. Run the chain $\mathcal{M}(\Gamma)$ initialized with $H^{(t)}$ for $\approx 10^6 \cdot n$ steps.

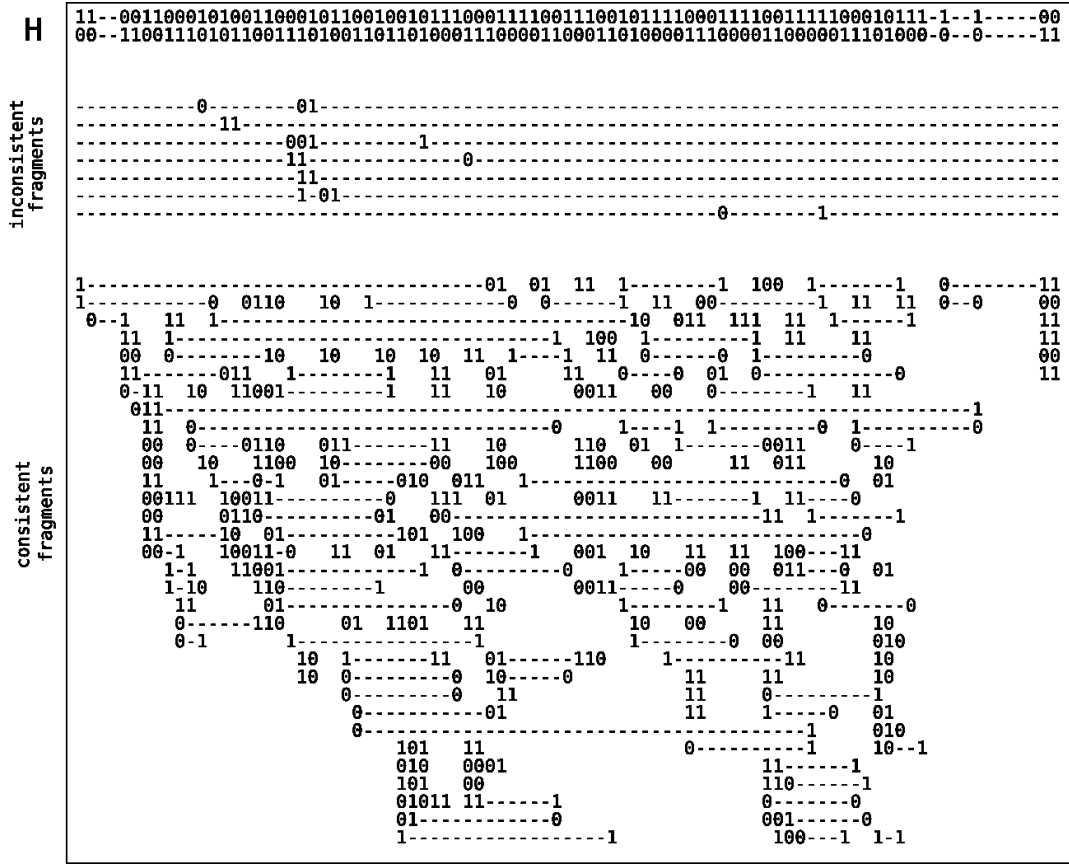


Figure 5.4: An example of a fragment matrix for a haplotype segment from chromosome 22 of HuRef. “Inconsistent fragments” (one on each line) correspond to fragments that are inconsistent with H . “Consistent fragments” (multiple independent fragments on each line, two independent fragments separated by whitespace) that perfectly match one of the two haplotypes in H . This example illustrates two common features of the HuRef data relevant for haplotype assembly: (i) the “gapped” nature of the fragment matrix, i.e. the presence of links between non-adjacent variants, (ii) haplotypes do not always link all variants that they span.

Steps 1-3 in the above algorithm represent an Γ determination phase where we start from a haplotype H^0 and Γ initialized to Γ_1 . We run the Markov chain for a certain number of steps ($c \cdot n$ where $c \approx 1000$) and then compute a new Γ using the current haplotype pair. This is repeated until we see no improvement in the likelihood of the best haplotype sampled by the Markov chain. After this initial Γ determination phase, we run the Markov chain initialized using the current haplotype and the final Γ for $\approx 10^6 \cdot n$ steps. The samples used to make inference about the posterior distribution are

drawn only from this Markov chain. For drawing samples from $\mathcal{M}(\Gamma)$, we discard the first $10000 \cdot n$ samples and thin the chain every $1000 \cdot n$ steps.

5.3 MEC score for haplotype assembly and posterior error probabilities

Given a haplotype assembly H , we would like to evaluate how “consistent” it is with the fragment matrix (sequenced reads) X . Each fragment represents a chunk of DNA from one of the two chromosomes and in the absence of sequencing errors, the alleles at variant sites covered by the fragment should perfectly match one of the two haplotypes. We define $\varepsilon_i[j, h] = 1$ if X_i and h disagree at position j , and let $\text{MEC}(X_i, h) = \sum_j \varepsilon_i[j, h]$ denote the number of alleles mis-matched between X_i and h .

For a haplotype pair $H = (h, \bar{h})$, let $0 \leq Z_i(H) \leq 1$ denote the probability that the fragment X_i is derived from the haplotype h . Define

$$\text{MEC}(X_i, H) = Z_i(H) \cdot \text{MEC}(X_i, h) + (1 - Z_i(H)) \cdot \text{MEC}(X_i, \bar{h})$$

The total MEC score is defined as the fraction of mismatched variant calls (Bafna et al., 2005; Rizzi et al., 2002)

$$\text{MEC}(X, H) = \frac{1}{n} \sum_i \text{MEC}(X_i, H)$$

The MEC score gives an estimate of the quality of the reconstructed haplotypes, i.e. lower this number, better the haplotypes. For a single haplotype pair H , we can set $Z_i(H) = 1$ if $\text{MEC}(X_i, h) < \text{MEC}(X_i, \bar{h})$ and 0 otherwise. On the other hand, if we are given a probability distribution π on the haplotypes and the matrix q of error probabilities, we can compute

$$Z_i(H) = \frac{\text{Pr}(X_i|q, h)}{\text{Pr}(X_i|q, h) + \text{Pr}(X_i|q, \bar{h})}$$

and the expected MEC score as

$$E_{\pi}(\text{MEC}(X)) = \sum_H \pi_H \text{MEC}(X, H)$$

Additionally, π can also be used to compute posterior error probabilities on the base call $X_i[j]$ as

$$\Pr(\varepsilon_i[j] = 1|\pi) = \sum_H \pi_H \{Z_i(H) \cdot \epsilon_i[j, h] + (1 - Z_i(H)) \cdot \epsilon_i[j, \bar{h}]\}$$

An MCMC algorithm that samples from the posterior distribution of haplotypes can be used to compute the posterior error probabilities where $\pi_H = Pr(X|H, q)$. These error probabilities represent probabilistic estimates of the reliability of each variant call, i.e. an error probability of 0.9 implies a 90% chance of the call being incorrect. To illustrate, consider the example of a fragment matrix with two columns and two fragments: 00 and 01. Let $q_i[j] = \hat{q} = 0.05$ be identical for all variant calls. The two haplotypes for this matrix are $H_1 = (00, 11)$ and $H_2 = (01, 10)$ with $\pi_{H_1} = \pi_{H_2} = 0.5$. The posterior error probability for each variant call can be easily computed to be ≈ 0.5 . Therefore, none of the variant calls is reliable and there is no information about the phase between the two variants present in the data.

5.4 Results

5.4.1 HuRef sequence data

The HuRef genome assembly (Levy et al. (Levy et al., 2007)) represents the sequence of a single human individual using traditional Sanger sequencing technology. It was derived from approximately 32 million reads and has a sequence coverage of 7.5. Using the HuRef sequenced reads and comparison between the HuRef genome assembly and the NCBI reference genomic sequence, a list of potential DNA variants was compiled. These variants are not restricted to single nucleotide polymorphisms, but also include short insertions/deletions, etc. The sequenced reads were mapped to the

HuRef assembly to determine the alleles at each variant. For each sequenced read, the sequencing quality values were used to assign a error probability for the variant sites. After applying various filters to define a set of reliable heterozygous variants, there were about 1.8 million heterozygous variants for the 22 autosomes (see (Levy et al., 2007) for details).

To illustrate the coverage and connectivity of the sequenced fragments, we present some statistics for chromosome 22, which has 24,967 heterozygous variants. For this chromosome, the fragment matrix had 103,356 rows where each row corresponds to a DNA fragment from one of the two copies of the chromosome. Hence, paired-end reads (sequenced ends of clones) are represented as a single row. 18,119 of these fragments correspond to such paired-end reads. About half of the fragments (53,279) link two or more variants and therefore are potentially useful for haplotype assembly. These 53,279 fragments correspond to 173,084 variant calls (≈ 7 calls per variant) in the fragment matrix. Using the overlap between these fragments, the chromosome can be partitioned into 609 disjoint haplotypes (in addition to 921 isolated variants) of varying lengths, the largest of which links 1008 variants. In terms of the actual physical distance spanned by haplotypes, the N50 haplotype length (length such that 50% of the variants are contained in haplotype segments of the given length or greater) is ≈ 350 kb. Note that a haplotype segment does not link all variants it spans (see Figure 5.4 for an illustration of a haplotype segment). Even if haplotype length is measured in terms of the number of variants linked, the N50 length is about 400 variants.

The importance of paired-end reads for haplotype assembly can be gauged from the comparison of the distribution of the number of variants among haplotypes of different sizes for a) reads including paired-end information vs b) unpaired reads (see Figure 5.5). If we ignore the paired ends, and split them into separate fragments, the linkage between the variants and consequently, the haplotype block sizes are greatly reduced. The number of disconnected haplotypes increases to 4378 with no haplotype having more than 100 variants.

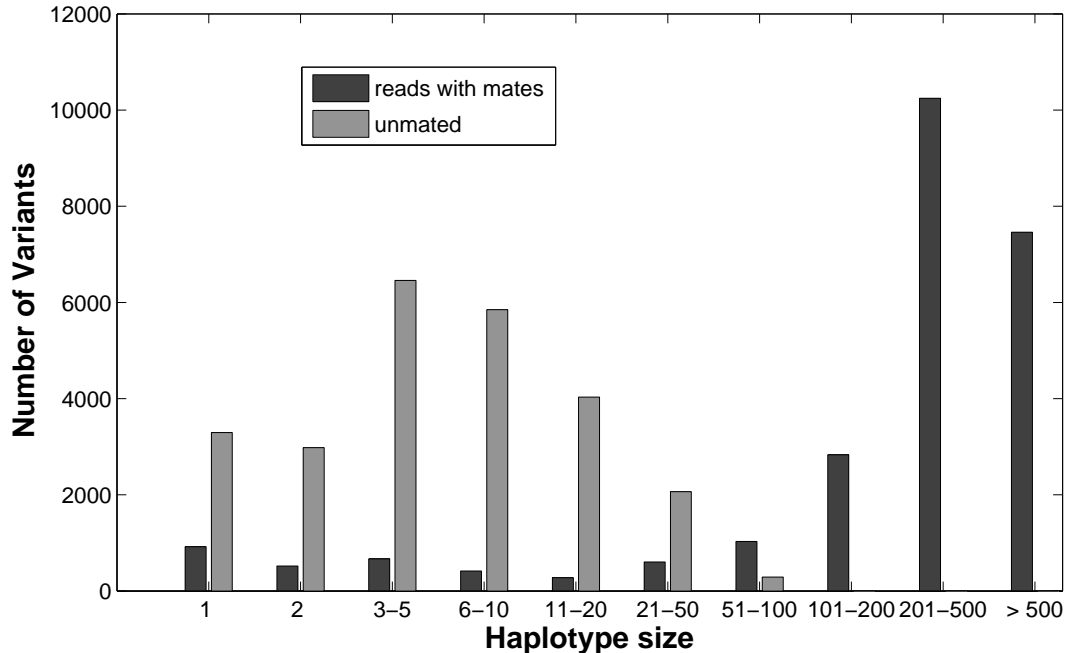


Figure 5.5: Distribution of the number of variants among haplotypes of different sizes (haplotype size is measured as the number of variants linked together in a haplotype) for chromosome 22 of HuRef. The y-axis is the aggregated number of variants that are part of haplotypes of a certain size. Haplotype size ‘1’ corresponds to isolated variants not connected to any other variant. The ‘reads with mates’ distribution corresponds to the complete fragment matrix. The ‘unmated’ distribution is obtained by splitting mate pairs into separate fragments.

5.4.2 Performance of HASH on Simulated data

To test the performance of HASH, we generated simulated data with varying error rates as follows: first, the fragment matrix X was modified to make it perfectly consistent with a particular haplotype. Next, to simulate an error rate of ε ($0 \leq \varepsilon \leq 0.1$), each variant call in the fragment matrix was “flipped” (changed from 0 to 1 or vice versa) independently with probability ε . For this modified fragment matrix, we know the true haplotypes and also the variant calls that are correct (those that were not flipped) and also those that are incorrect (the ones that were flipped during simulations). Therefore, we can assess the performance using two different criteria: a) the distance of the reconstructed haplotypes from the true haplotypes and b) the ability to predict which

variant calls are incorrect.

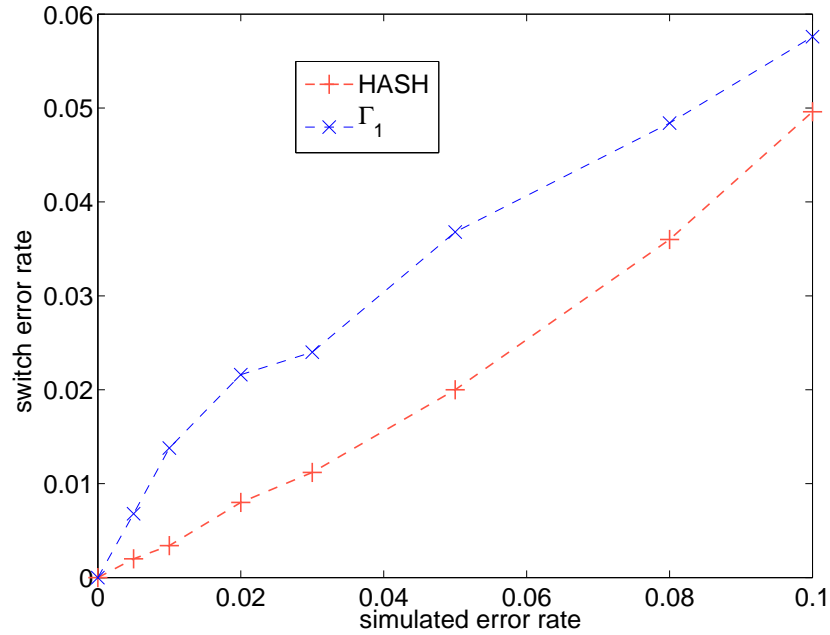


Figure 5.6: Comparison of the switch error rate for the algorithm HASH and the MCMC algorithm with Γ_1 . The y-axis is the average switch distance of the reconstructed haplotypes from the true haplotypes. The x-axis (simulated error rate) is the fraction of variant calls in the fragment matrix that were flipped.

In Figure 5.6, we plot the average switch distance of the Maximum Likelihood reconstructed haplotypes from the true haplotypes as a function of ε . Average switch distance or switch error rate (Lin et al., 2002) is defined as the fraction of positions for which the phase between the two haplotypes is different relative to the previous position. The switch error rate increases roughly linearly with increasing error rate and is ($\sim 2\times$) lower for HASH than for the MCMC algorithm with Γ_1 . This is expected given the slow convergence of the Markov chain with Γ_1 . The switch error rate for the greedy heuristic (Levy et al., 2007) is also high in comparison with HASH (results not shown).

Using an MCMC procedure, one can estimate the posterior error probability for each variant call in the fragment matrix. Given a haplotype pair $H = (h, \bar{h})$, let $Z_i(H)$ denote the probability that fragment X_i is sampled from h . Denote $\varepsilon_i[j, h] = 1$ if X_i and h disagree at position j . Finally, let $\varepsilon_i[j] = 1$ to denote that $X_i[j]$ is called incorrectly,

and $\varepsilon_i[j] = 0$ otherwise. Then, the posterior error probability can be computed as follows:

$$\Pr(\varepsilon_i[j] = 1|\pi) = \sum_H \pi_H \{Z_i(H) \cdot \varepsilon_i[j, h] + (1 - Z_i(H)) \cdot \varepsilon_i[j, \bar{h}]\}$$

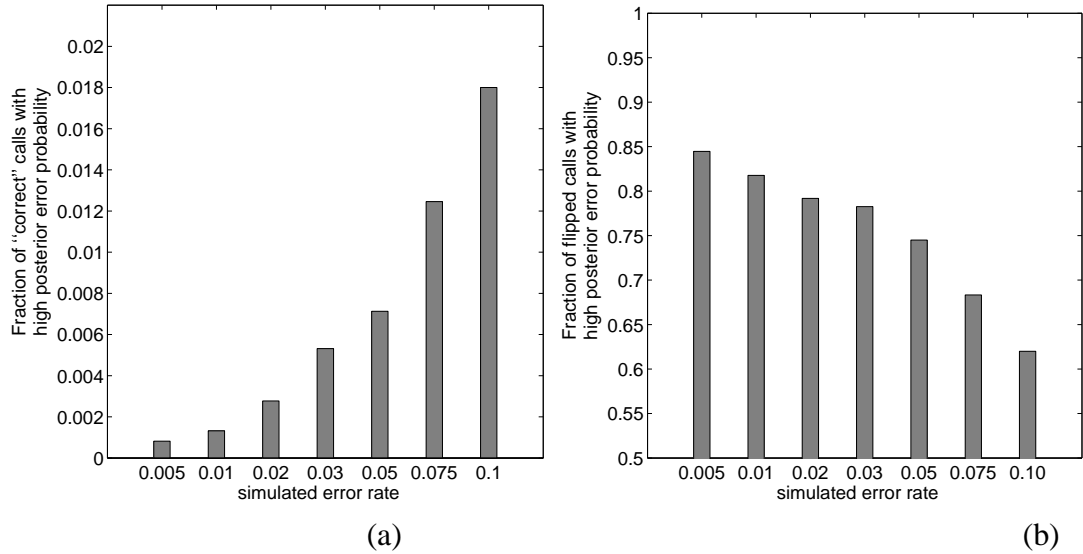


Figure 5.7: Fraction of variant calls with a posterior error probability ≥ 0.5 using the HASH algorithm for different values of ε . (a) *False-positive rate*, given by the fraction of “correct” variant calls with high posterior error probabilities. (b) *True-positive rate*, given as the fraction of “flipped” variant calls with high posterior error-probability.

Here $\pi_H = Pr(X|H, q)$ is a probability distribution over H . See Supplementary Methods for a complete description. We compare the posterior error probability for the “correct” variant calls with those for the “incorrect” variant calls to demonstrate that our algorithm HASH can predict the incorrect variant calls. In Figure 5.7(a), we plot the false positive rate (fraction of correct variant calls that had a posterior error probability greater than 0.5) for different values of ε . For an error rate of 0.02 (typical of the HuRef sequence data, see Figure 5.9), the fraction of incorrect variant calls with a high posterior error probability (> 0.5) is almost 80%. In Figure 5.7(b), we plot the true positive rate (fraction of flipped base calls that had a posterior error probability greater than 0.5). Increasing the cutoff value for the posterior error probability reduces both the true

positive rate and the false positive rate. For an error rate of 0.02, 65% of the incorrect (or flipped) variant calls have a posterior error probability greater than 0.95 while only 0.015% of the correct variant calls have such a high posterior error probability.

The plots suggest that the error in reconstruction is very low for typical sequencing errors, but increases with increasing error rate. Also, our measure for estimating accuracy is (perhaps, overtly) conservative. For example, if there is a single call for a variant and this variant call is flipped, it is not possible to reconstruct the true haplotype or predict that this variant call is incorrect. Flipping a variant call not only affects the posterior error probability of that variant call, but the error probability of variant calls that cover the same column. Therefore, increasing the error rate is expected to increase the number of ‘correct’ variant calls with a high posterior error probability. Also, if the error rate is large and the number of fragments covering each variant is small, it may not be possible to reconstruct the true haplotype exactly from the mutated fragment matrix.

5.4.3 HASH versus other MCMC algorithms

Our goal in devising HASH is to enable the Markov chain to move out of local optima and transition to haplotypes with greater likelihood. We compared the performance of HASH against two other MCMC algorithms: i) $\mathcal{M}(\Gamma_1)$, the Markov chain with Γ_1 and ii) $\mathcal{M}(\Gamma)$ where Γ was computed once using the recursive graph-partitioning on $G(X)$. Recall that HASH is similar to (ii) except that Γ is updated iteratively. For this, we used data from chromosome 22 and looked at the maximum-likelihood haplotype pair sampled by each algorithm. The results shown are for a block with ≈ 200 columns from chromosome 22 (see Figure 5.8(a)). In each case, the Markov chain was initialized with a random haplotype pair. As expected, HASH dominates both in the likelihood of the sampled solution, and in the speed with which the solution is reached. $\mathcal{M}(\Gamma_1)$ gets stuck in a local optima and will take a prohibitively large number of steps to sample the maximum likelihood solution.

In Figure 5.8(b) we zoom in on the ‘ Γ update’ phase of the HASH algorithm

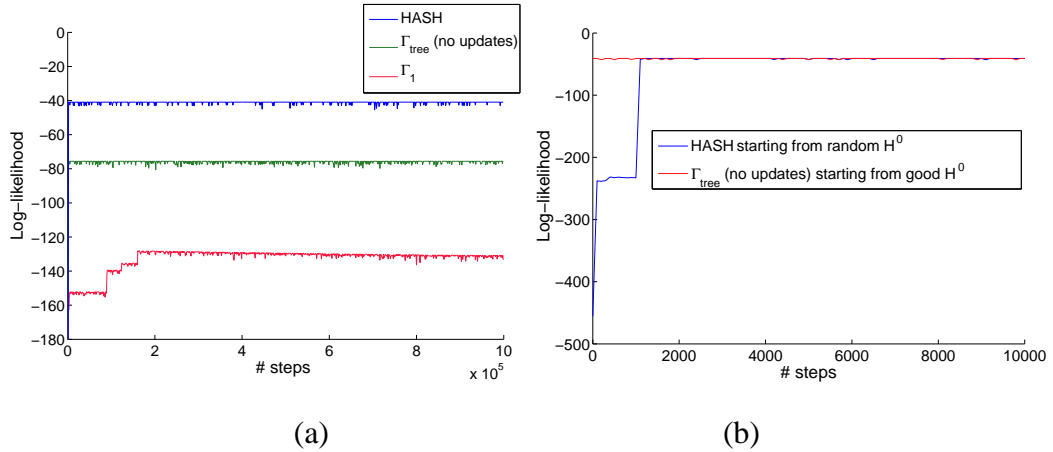


Figure 5.8: Results of running the MCMC algorithm with different Γ on a fragment matrix with $n = 200$ columns (from chromosome 22 of HuRef genome). a) A comparison of the HASH algorithm against two other MCMC algorithms: i) $\mathcal{M}(\Gamma_1)$, ii) $\mathcal{M}(\Gamma)$ where Γ was computed using the recursive graph-partitioning algorithm $G(X)$. All algorithms were initialized with a random haplotype pair. b) Comparison of HASH algorithm initialized with a random haplotype vs $\mathcal{M}(\Gamma)$ (graph-partitioning) started with a good haplotype. Note that we are zooming in on the first 10K steps in the iteration.

for the above example. The HASH algorithm was initialized with a completely random haplotype. We observe that the likelihood of the best haplotype sampled by the HASH algorithm after a few updates to Γ is identical to that of the Markov chain with the graph-partitioning based Γ started from a good quality solution. Although the results shown in Figure 5.8 are for one particular example, they are similar for all data-sets (data not shown). The two results combined show that the sample space has many locally optimal solutions that one could be trapped in, but dynamic updates to the Markov chain architecture, as described by HASH allows for rapid convergence, increasing the likelihood of sampling the globally optimum solution.

5.4.4 Haplotypes for HuRef

We compared the most likely haplotype assembly obtained using HASH with the greedy haplotype assembly (Levy et al., 2007) for each of the 22 autosomes of the HuRef individual. HASH was run independently on each of the disjoint haplotype blocks for

a chromosome. For each chromosome, we compared the haplotype assembly against the fragment matrix and computed the MEC (Minimum Error Correction) score (Bafna et al., 2005), defined as the minimum number of variant calls in the fragment matrix that need to be modified for every fragment to perfectly match one of the two haplotypes. The MEC score represents a parsimonious estimate of the discordance between the haplotypes and the fragment matrix. A more detailed formulation of the MEC score is given in the Supplementary methods. In Figure 5.9, we compare the MEC scores for three different methods: Greedy heuristic (Levy et al., 2007), MCMC algorithm with Γ_1 and HASH. The haplotype assembly derived using HASH has a lower MEC score for each chromosome, reflecting the greater accuracy of the haplotypes. For chromosome 22, the MEC score for HASH was 20% lower than the greedy algorithm. Note that the MEC score is not expected to be zero, even for the true haplotypes, due to errors in base-calling.

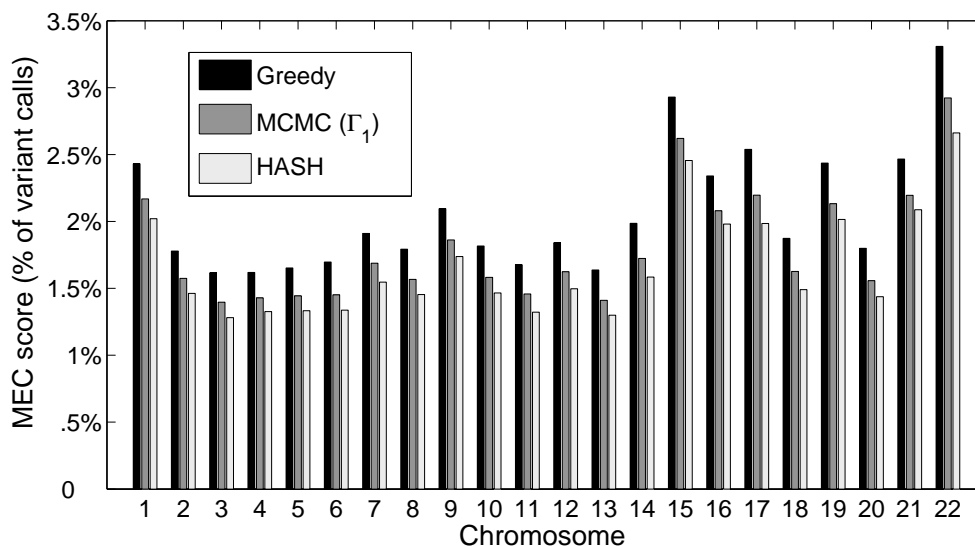


Figure 5.9: The percentage of variant calls that are inconsistent with the best haplotype assembly for three different methods: Greedy heuristic (Levy et al., 2007), MCMC algorithm with Γ_1 and the HASH algorithm for the 22 autosomes of HuRef.

We also compared the log likelihood of the haplotype assemblies for the greedy algorithm and HASH. The log-likelihood was computed using the sequencing quality

values to estimate the q matrix. We found that the log likelihood for the haplotypes reconstructed using HASH was consistently higher than that of the greedy haplotypes indicating that the haplotypes are significantly more accurate. For example, the log-likelihood of the greedy haplotype assembly for chromosome 22 (summed over all disjoint haplotypes) was -15683.4. In comparison, the most likely haplotype assembly using the HASH algorithm had a log-likelihood of -11944.25 (a reduction of 23.8%).

We compared the posterior error probabilities for each variant call against the sequencing quality values. To allow an unbiased comparison, the HASH algorithm was run using uniform error probabilities estimated from the greedy haplotypes (\hat{q} = fraction of inconsistent variant calls). For chromosome 22, 3919 of the 173804 variant calls had a posterior error probability greater than 0.5. If we restrict the comparison to variant calls with a sequencing quality value below 20 ($q \geq 0.01$), 1203 out of 28532 such variant calls had a high posterior error probability. This represents a 2-fold enrichment of “erroneous” variant calls in the tail of quality value distribution. In Figure 5.10, we can see that the fraction of variant calls with a high posterior probability increases with increase in the error probability (or decreasing sequencing quality values). This correlation between high posterior error probabilities and low sequencing quality values represents an independent confirmation of the quality of the reconstructed haplotypes and also indicates that some of the inconsistencies between the reconstructed haplotypes and the fragments are a result of sequencing error.

5.4.5 Estimating accuracy of HuRef haplotypes

The HuRef haplotypes obtained using HASH are highly consistent with the sequenced fragments and have a low MEC error rate (see Figure 5.9). However, we also want to be able to estimate the absolute accuracy of the HuRef haplotypes. The absolute accuracy can be expressed in terms of the “switch error rate”(Lin et al., 2002) or the fraction of adjacent pairs of variants whose phase in the HuRef haplotypes is incorrect. We have computed two independent estimates of the switch error rate; one based on the

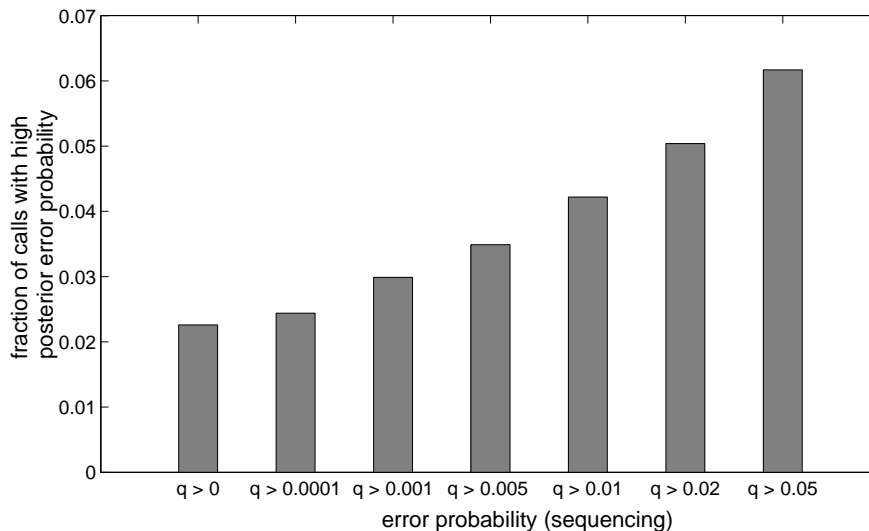


Figure 5.10: Fraction of variant calls with high posterior error probability (≥ 0.5) for different values of the error probability q (derived from the sequencing quality values) for chromosome 22 of HuRef.

haplotypes samples generated by our MCMC algorithm and another through comparison of the HuRef haplotypes to the population haplotypes from the HapMap project.

Switch error estimates using samples from the MCMC algorithm We used the haplotypes sampled by the algorithm HASH to estimate the reliability of the phase between adjacent pairs of variants in a haplotype segment. For a pair of adjacent variants (i, j) , if we denote the two alleles at each site by 0 and 1, there are two possible haplotype pairs: (00, 11) and (01, 10). Based on haplotypes sampled by the Markov chain, the switch error probability for a pair (i, j) was estimated as the fraction of times the less frequent haplotype pair was observed. See Figure 5.4 for a plot of switch error probabilities for a haplotype segment from HuRef. The switch error rate for a chromosome can be approximated as the average of the switch error probabilities for adjacent pairs. For chromosome 22 of HuRef, the switch error rate was estimated to be 0.009 using 1000 samples.

Switch error rate based on comparison to HapMap haplotypes One of the benefits of inferring haplotypes from sequence data is that the local accuracy of the haplotypes is unlikely to be affected by the level of Linkage Disequilibrium in a region. This also presents the opportunity of using LD in population data to detect switch errors in the HuRef haplotypes. For a pair of variants that are in strong LD in population data, the correct HuRef phasing is expected to match the more likely population based phasing. If the inferred HuRef phasing does not match the preferred population phasing, one can infer a switch error with some probability (the probability value depends upon the strength of LD between the pair of variants). We use this idea to empirically estimate the switch error rate of the HuRef haplotypes. As the HuRef individual is of Caucasian origin, we have used the haplotypes from the CEU population in the HapMap project for this comparison. We identified the subset of SNP variants in HuRef that were also genotyped in the HapMap project. For each pair of adjacent SNPs in this subset, there are two possible haplotype phasings: (00, 11) and (01, 10). Let f_{00} , f_{11} , f_{01} and f_{10} represent the frequencies of the four haplotype pairs in the HapMap CEU sample. If $(f_{00} \cdot f_{11}) > (f_{01} \cdot f_{10})$, the pair (00, 11) is defined to be the preferred HapMap phasing. Otherwise, (01, 10) is the preferred HapMap phasing. For a pair of adjacent HapMap SNPs in the HuRef haplotypes (that were part of the same haplotype segment), the phasing of the HuRef individual is compared to the preferred HapMap phasing for that pair. The mismatch rate is defined as the fraction of pairs for which the HuRef phasing does not match the preferred HapMap phasing. In Figure 5.12, we plot the mismatch rate of the HuRef haplotypes for chromosome 22 (estimated using HASH) as a function of LD (measured using r^2). The mismatch rate is lowest for pairs with high levels of LD (0.008 for pairs with $r^2 > 0.8$) and increases to 0.031 for all pairs. The mismatch rate for pairs with high levels of LD can mainly be attributed to switch errors in the HuRef haplotypes. For pairs of SNPs with low LD, mismatches between the HuRef haplotypes and the preferred HapMap phasing can represent switch errors or chance mismatches (see Figure 5.11 for an illustration).

To correctly estimate the error rate, we first compute an expected mismatch rate

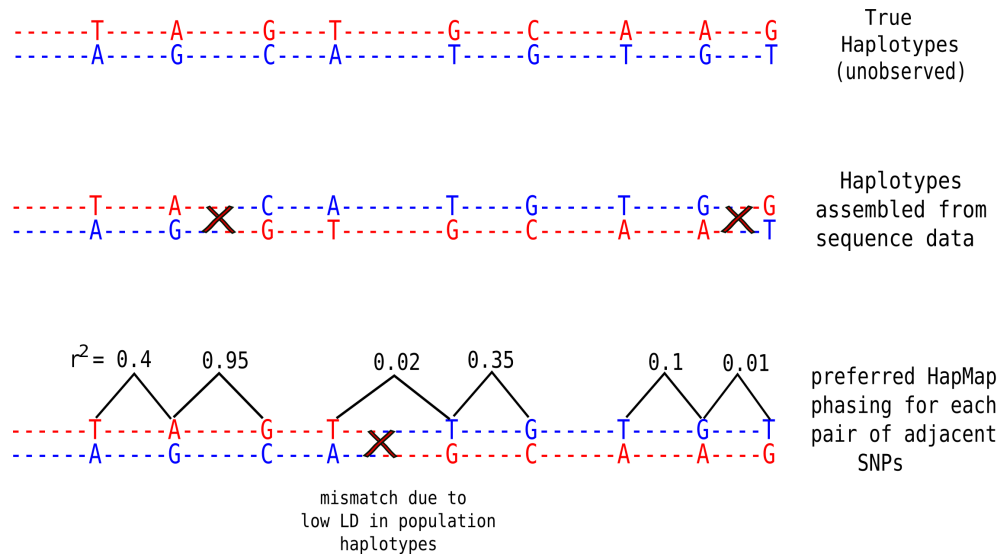


Figure 5.11: Comparison of haplotypes assembled using sequence data with the preferred HapMap phasing for each pair of adjacent SNPs inferred from the HapMap haplotypes. For three pair of adjacent SNPs, the phase of the sequence-based haplotypes mismatches the preferred HapMap phasing (indicated by crosses). The first pair shows strong linkage disequilibrium ($r^2 = 0.95$) and therefore the mismatch is more likely to represent a switch error in the sequence-based haplotypes. For the second pair of SNPs, the sequence based haplotypes are correct and the mismatch is due to low LD between the SNP pair. For the third pair, LD is low and the mismatch is due to a switch error in the sequence-based haplotypes.

for the HapMap haplotypes as follows: for every pair of adjacent SNPs, we sample one of the two haplotype pairs ((00, 11) or (01, 10)) based on the haplotype frequencies in the HapMap haplotypes. The expected mismatch rate is the fraction of pairs for which the sampled pair mismatches the preferred HapMap phasing. For a particular value of r^2 , we define the “adjusted mismatch rate” as the mismatch rate minus the expected mismatch rate. The adjusted mismatch rate represents an estimate of the switch error rate of the HuRef haplotypes that is adjusted for variation in LD in the HapMap haplotypes. In Figure 5.12, we plot the “adjusted mismatch rate” for HASH and for the greedy heuristic. We observe that the adjusted mismatch rate for HASH is nearly independent of LD, ranging from 0.011 for all pairs to 0.0078 for pairs of SNPs with $r^2 > 0.8$. The adjusted mismatch rate for the greedy heuristic is almost three times that of HASH,

providing the strongest proof of the greater accuracy of the haplotypes inferred using HASH.

Both internal and external estimates indicate that the switch error rate of the HuRef haplotype assembly is about 0.01. The switch error rate for HapMap individuals from the CEU and YRI samples has been estimated to be 0.0053 and 0.0216 respectively (Marchini et al., 2006). The haplotypes for these individuals have been inferred using a combination of trio and population information. The increased error-rate for YRI is due to lower levels of LD in the Yoruban population. Switch error rates for haplotypes inferred without trio information are typically much higher (0.054 for CEU individuals). An advantage of inferring haplotypes using sequence data is that the error rates are expected to be independent of the ancestry of the individual. Moreover, since the switch errors are distributed independent of LD, the error rate could be reduced further by incorporating LD information from population data in the haplotype assembly.

5.5 Discussion

With the rapid development of new sequencing technologies (Bentley, 2006) comes the promise of individualized sequencing, wherein the complete genomic sequence of individuals will be available. As shown by Levy et al. (Levy et al., 2007), individual sequencing in the presence of paired-ends allows one to reconstruct accurate long haplotypes using a simple method. There are two challenges to sequence-based haplotype inference: the high cost of whole genome shotgun sequencing at a reasonable sequence coverage using Sanger sequencing and the feasibility of haplotype assembly using other sequencing methods. In the past few years, next-generation sequencing technologies have drastically reduced the cost of sequencing complete genomes. Additionally, some of these technologies have the ability to generate paired-end sequences which is critical for haplotype assembly. Haplotype assembly is feasible when the sequenced fragments are long enough to cover multiple variants, and the sequence coverage is high enough to overcome base-calling error. Although the read lengths for these methods

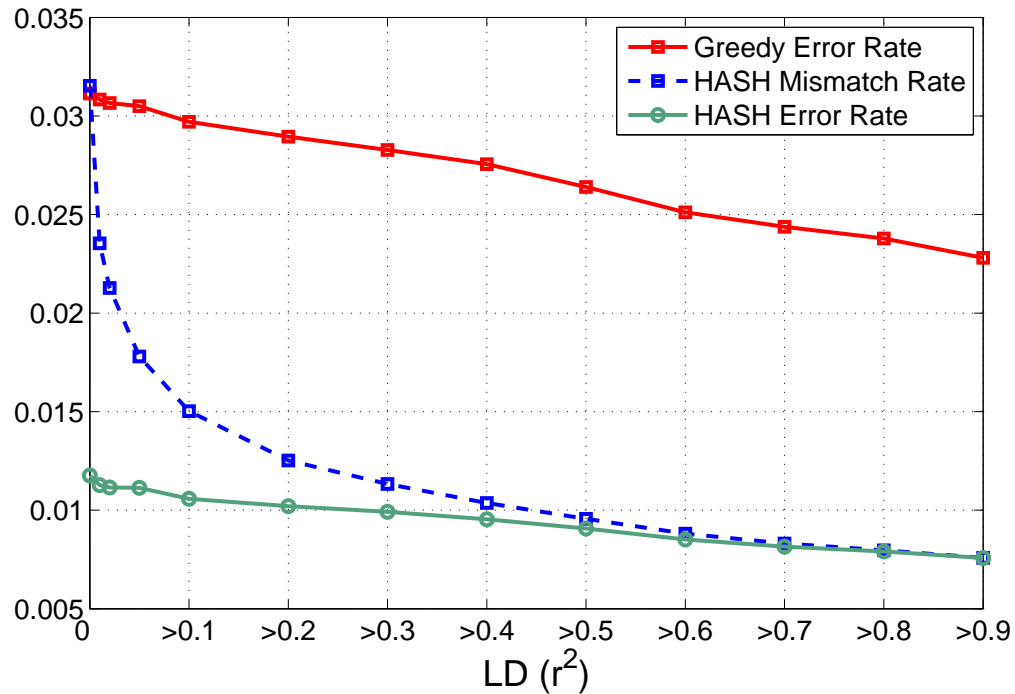


Figure 5.12: Mismatch Rate and the “Adjusted Mismatch Rate” (Error Rate) of the HuRef haplotypes estimated by comparison with the CEU HapMap haplotypes. The error rate is plotted as a function of r^2 , i.e. computed for all pairs of adjacent SNPs with r^2 greater than a certain value.

are much shorter than traditional Sanger sequencing, continued enhancements in these technologies will make haplotype assembly feasible in the near future.

Haplotype assembly from sequenced reads of an individual genome has several advantages over haplotypes obtained by computationally phasing SNP genotypes from a population. First, the accuracy of the phasing is not limited by regions of low linkage disequilibrium and it is possible to recover very long haplotypes spanning several hundred kilobases. Second, it is possible to assemble “complete” haplotypes linking alleles at all variants such as SNPs, insertion/deletions, etc that are heterozygous in the individual. Third, the accuracy of haplotypes inferred from genotype data depends a great deal on the knowledge of ancestry of the individual, while haplotype assembly from sequence data does not require knowledge of the population of origin of the individual. It is important to note that these two approaches for inferring haplotypes are

complementary to each other. As individual genomes are sequenced, population data could be combined with sequence data to obtain longer and more accurate haplotypes for an individual. Linkage Disequilibrium from population data could be used to determine the phase between variants that are not linked by sequenced reads, while sequence data could be used to infer haplotypes across regions of low LD. The highly accurate haplotypes generated by the HapMap project for the CEU and YRI samples could prove especially useful for improving the quality of haplotypes assembled using individual sequencing.

We have described a Markov chain Monte Carlo algorithm for haplotype assembly that samples haplotypes given a list of all heterozygous variants and a set of sequenced reads mapped to a genome assembly. Our emphasis has been on describing how a particular choice of moves for the Markov chain enables it to sample the haplotype space more efficiently than a naive Markov chain. We have shown that haplotypes reconstructed using HASH are much more consistent with the sequenced reads than haplotypes inferred using a greedy heuristic. Comparison of the HuRef haplotypes to the HapMap haplotype data suggests that the error rate of haplotype reconstruction using HASH is low ($\sim 1.1\%$), and independent of the local recombination rate. Instead, simulations show that the error rate depends upon the sequencing error and depth of coverage. As technologies improve, the cost and error rates will improve further, increasing the power and accuracy of haplotype assembly.

There are some aspects of haplotype assembly that could be investigated further. In our approach, we assume that the list of variants is compiled in advance using the sequenced reads and haplotype assembly is performed using this list. Detection of SNPs and variant sites from sequencing data is a challenging problem in itself and one can possibly integrate the variant detection phase with the estimation of haplotypes. This approach has recently been adopted by Kim et al. (Kim et al., 2007) and can have certain advantages for genomes whose variant sites are not well characterized. Our model for haplotype likelihood considers each variant call independently. One can incorporate more complex error models where all the variant calls for a read are erroneous e.g. as a

result of the read being incorrectly mapped, or some of the variant sites do not represent real polymorphic variants, e.g. paralogous SNPs. The HASH framework is independent of the likelihood model and can be easily adapted for such models.

Finally, we note that there are some novel aspects of our Markov chain Monte Carlo algorithm, HASH. We have shown, both empirically and theoretically, that a simple Markov chain with local moves, i.e. a chain in which all transitions are between haplotypes that differ in a single column, is unable to sample the haplotype space efficiently. We have proposed a Markov chain with non-local moves that allows transition between haplotypes that differ in multiple columns. The transition matrix of this Markov chain is determined by min-cut computations on an associated graph derived from the sequenced reads. Moreover, the Markov chain architecture is dynamically updated periodically, to escape local minima.

5.6 Acknowledgement

Chapter 5, in full, will be published in Genome Research, V. Bansal, A. Halpern, N. Axelrod, V. Bafna, “An MCMC Algorithm for Haplotype Assembly from Whole-genome Sequence Data”. The dissertation author was the primary investigator and author of this paper.

Chapter 6

A Combinatorial Algorithm for the Haplotype Assembly Problem

6.1 Introduction

In the previous chapter, we described HASH, a Markov Chain Monte Carlo (MCMC) algorithm for the haplotype assembly problem and demonstrated that the HuRef haplotypes based on HASH were much more accurate than those using the greedy heuristic. In this chapter, we describe a novel combinatorial approach for the Haplotype Assembly problem based on a problem related to the MAX-CUT problem. Our algorithm HapCUT tries to minimize the MEC score of the reconstructed haplotypes by iteratively computing max-cuts in graphs derived from the sequenced fragments. Our algorithm is motivated by the HuRef sequence data and is applicable to sequenced fragments of any length with an arbitrary number of gaps. Using the HuRef sequence data, we demonstrate that our algorithm is significantly more accurate than the greedy heuristic of Levy et al., 2007. We also compare the performance of HapCUT with a previously proposed heuristic for this problem, namely Fast Hare (Panconesi and Sozio, 2004), and find that our algorithm consistently outperforms this heuristic. The MEC score of the haplotypes reconstructed using HapCUT is comparable to those using

a Markov chain Monte Carlo algorithm while being much faster to compute. While the problem of optimizing the MEC score is NP-hard even for gap-less fragments (Cilibrasi et al., 2005), and unlikely to admit an efficient algorithm, HapCUT represents a fast and accurate heuristic for haplotype assembly using real sequence data. We will also describe a Maximum Likelihood based estimator of the switch error rate of the HuRef haplotypes based on the CEU HapMap haplotypes and show that the HuRef haplotypes inferred using HapCUT have a low switch error rate (1.1 – 1.4%).

6.2 Methods

6.2.1 Preliminaries and Optimization

The genome sequence assembly for a chromosome is a mix of the two haploid chromosomes. If we align all of the fragments to the assembly, certain sites (*columns* in the alignment) will show identical values (homozygous) for all fragments, while others will have different values (heterozygous) for different fragments. Note that heterozygous sites in the alignment could correspond to a single base pair (SNPs) or multiple base pairs, e.g. deletion/insertion variant. Sites that are homozygous are discarded, as they are not useful for haplotype phasing. Likewise, all sites with more than 2 alleles are discarded, as all variant sites should be bi-allelic for a diploid genome. Arbitrarily re-labeling the variant alleles as 0 and 1, the input data can be represented as a ternary matrix X of size $m \times n$, where m is the number of fragments and n is the number of heterozygous sites. The i -th fragment (row i of X) is described by a ternary string $X_i \in \{0, 1, -\}^n$, where $'-'$ corresponds to the variant loci not covered by the fragment. The objective of haplotype assembly is to reconstruct the two haplotypes, i.e. determine the combination of alleles present on a single chromosome at the heterozygous sites. The haplotypes can be represented as an unordered pair $H = (h_1, h_2)$ of binary strings, each of length n . Since we only consider sites that are heterozygous in the individual genome, h_2 is constrained to be the bit-wise complement of h_1 .

In the absence of any errors, the rows of the fragment matrix can be partitioned into two disjoint sets such that every column is homozygous in each set (Lancia et al., 2001). Further, the consensus values can be used to reconstruct the two haplotypes. However, such a perfect bi-partition is not possible when there are errors in the fragment matrix. In the presence of errors, the objective of haplotype assembly is to find a bi-partition or a pair of haplotypes that minimizes some objective function. Under the MEC (minimum error correction) criterion, the objective is to change the smallest number of entries in the fragment matrix such that the resulting matrix admits a perfect bi-partition. The MEC objective is also equivalent to finding a pair of haplotypes H for which the MEC score of the fragment matrix $MEC(X, H)$ is minimum.

If $d(X_i, h)$ denotes the number of mismatches between the fragment X_i and haplotype h (ignoring the '-' in X_i), then

$$MEC(X_i, H) = \min\{d(X_i, h), d(X_i, \bar{h})\}$$

and the overall score is given by

$$MEC(X, H) = \sum_i MEC(X_i, H)$$

This leads to the natural parsimony based optimization problem of computing haplotypes with minimum *MEC* score. For notational convenience, we will denote the error for a haplotype pair H as $MEC(H)$, whenever X is implicit. We will focus on designing an algorithm for the MEC objective function. Other objective functions for haplotype assembly have previously been proposed, such as MFR (Minimum Fragment Removal) where the objective is to remove the smallest number of fragments to make the fragment matrix error free, and MSR (Minimum SNP Removal) which models false variant sites. However, the simplest (and most-common) source of error is due to base-miscalling and the MEC objective serves as a good model for this type of error. Moreover, MEC can also indirectly model other sources of error, e.g. a haplotype assembly with a low MEC score is also likely to be good under the MFR objective.

6.2.2 MEC optimization using max-cuts

Among the various objective functions for the haplotype assembly problem, MEC seems to be the most difficult to optimize. While MFR and MSR can be solved optimally for gap-less fragments, finding the optimal MEC solution has been shown to be NP-hard even for gap-less fragments (Lippert et al., 2002; Cilibrasi et al., 2005). MEC has been shown to be $O(\log n)$ approximable in the general case by Panconesi and Sauzio (Panconesi and Sozio, 2004), who also describe a heuristic Fast Hare for the problem. Here, we provide a formulation for MEC optimization based on graph-cuts, which leads to a simple but effective algorithm.

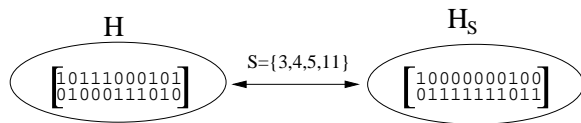
Given a fragment matrix X , and a haplotype pair H , we define the graph $G_X(H)$, with vertices defined by columns of X . We abuse notation slightly by referring to the vertex set as X . The set E_H of edges of this graph is defined by pairs of columns that are linked by some fragment. Let $X_i[j, k]$ and $H[j, k]$ represent the fragment i and haplotype pair H respectively, when restricted to the pair of columns (j, k) . The weight of the edge $(j, k) \in E_H$ connecting columns j, k is defined as

$$w_H(j, k) = \|\{i \mid MEC(X_i[j, k], H[j, k]) = 1\}\| - \|\{i \mid MEC(X_i[j, k], H[j, k]) = 0\}\|$$

Informally, the weight of the edge (j, k) is the number of fragments inconsistent with the current phase between the pair minus the number of fragments consistent with the phase $H[j, k]$. In other words, the weight represents the “weakness” of the phasing between columns i and j . A *Cut* in the graph $G_X(H)$ is defined by a subset $S \subseteq X$ of vertices. The weight of a cut S in $G_X(H)$ is given by

$$w_H(S) = \sum_{j \in S, k \in X-S} w_H(j, k)$$

Given a haplotype pair H , and a cut S in $G_X(H)$, the haplotype obtained by flipping the values of the columns in S is denoted by H_S , as illustrated in the example below:



Such transformations are effective for improving the MEC score, as exemplified by the following theorem.

Theorem 20: Let X be a fragment matrix with each fragment of length 2. For any haplotype pair H , let $S \subseteq X$ be a positive weighted cut $w_H(S) > 0$ in the graph $G_X(H)$. Then

$$\text{MEC}(H_S) = \text{MEC}(H) - w_H(S) < \text{MEC}(H)$$

If S is a MAX-CUT in the graph $G_X(H)$, then H_S is an optimal MEC solution for X .

Proof: Consider a cut S in the graph $G_X(H)$. Let H_S be the haplotype obtained by flipping the columns S of H . Clearly, $\text{MEC}(H) - \text{MEC}(H_S)$ is equal to the value of this cut $w_H(S)$. The maximum value of this difference is reached when the cut is a max-cut and therefore H_S is an optimal MEC solution. ♣

The above theorem implies that given a current haplotype pair H , any positive weight ($w_H(S) > 0$) cut leads to a haplotype (H_S) with a lower MEC score. This can be repeated iteratively resulting in haplotypes with decreasing MEC scores. If we can compute the MAX-CUT in a single step, we can find the optimal MEC solution, however this is not necessary. Based on this observation, an algorithm that looks for positive cuts in $G_X(H)$ can be used to optimize the MEC score as below.

Procedure HapCUT

Initialization: Choose an initial haplotype configuration H^1 randomly.

Iteration: For $t = 1, 2, \dots$

1. Construct the graph $G_X(H^t)$

2. Compute a cut S in $G_X(H^t)$ such that $w_H(S) \geq 0$
3. If $\text{MEC}(H_S^t) \leq \text{MEC}(H^t)$, $H^{t+1} = H_S^t$
4. Else $H^{t+1} = H^t$

The iterative procedure HapCUT is run until we can no longer get an improvement in the MEC score. While Theorem 20 holds only when all fragments have length 2 (and also for fragments of length 3 as we show in the next section), the algorithm HapCUT as described above works for arbitrary sized fragments. In order to ensure that HapCUT has good performance on real data, it is important to be able to compute high-scoring cuts in the graph $G_X(H)$. First, we show how the edge weights of the graph $G_X(H)$ can be weighted appropriately for long fragments.

6.2.3 Assigning weights to edges of $G_X(H)$

In the previous section, we described a simple formula to assign a weight to each edge of the graph $G_X(H)$. This formula gives disproportionately more weight to longer fragments, i.e. a fragment of length k contributes a total absolute weight of $\binom{k}{2}$ to the graph. The weighting scheme can be modified to ensure that Theorem 20 also holds for fragments of length 3. We simply scale the contribution of the fragment to each edge by $1/2$. Now, a fragment of length 3 can have a MEC of 0 or 1. An MEC of 0 corresponds to the three vertices (columns of the fragment) being on the same side of the cut and therefore contributing 0 to the cut value. An MEC of 1 corresponds to two of the vertices being on one side and therefore the fragment contributes exactly 1 to the cut after scaling. As the scaling of $\frac{1}{k-1}$ for a fragment of length k is consistent with the weights for fragments of length 2 and 3, we adopt it for computing the edge weights of arbitrary length fragments. Results on real data indicate that this works well, even though we do not know of a scaling under which Theorem 20 holds for arbitrary length fragments.

6.2.4 Computing Max-Cuts

The problem of computing a maximum-weighted cut is known to be NP-complete (Garey and Johnson, 1979), even when all edge weights are restricted to be 1. However, we only need to find a positive-weighted cut in order to improve the MEC score. Simple heuristics can find good cuts if all weights are positive. Indeed, a greedy heuristic (Sahni and Gonzalez, 1974) will give a cut which has at least 0.5 of the total weight of the edges of the graph. When the MEC score of H is poor and far away from the optimal MEC value (e.g. for a random haplotype pair), most of the edges of the graph $G_X(H)$ have positive weights and finding a positive-weighted cut is easy. However, when the MEC score of H is close to the optimum, most of the edges of the graph $G_X(H)$ have negative weights, and the greedy algorithm is not guaranteed to find a positive-weight cut. On the other hand, presence of a highly negative weight edge between two vertices s and t of the graph also implies that a positive-weight cut is unlikely to separate s and t . Therefore, for the purpose of computing a positive-weight cut, we can “contract” the edge (s, t) . We use a two-step greedy algorithm for computing a max-cut in $G_X(H)$. First, we find a cut where most of the negative weight edges do not go across the cut. In the second step, we move vertices from one side of the cut to the other if this improves the weight of the cut. The complete greedy heuristic is described below:

Greedy-Cut($G_X(H)$)

Initialization: BestCut = $-\infty$

Iteration: Iterate $O(m \log m)$ times

1. Chose an edge (s, t) of the graph uniformly at random
2. Initialize $S_1 = \{s\}$ and $S_2 = \{t\}$
3. While $S_1 \cup S_2 \neq V$
 - (a) For each vertex $v \notin S_1 \cup S_2$ compute the score $A(v) = \sum_{s_1 \in S_1} w_H(v, s_1) - \sum_{s_2 \in S_2} w_H(v, s_2)$

- (b) Let v_{max} be the vertex for which $|A(v)|$ is maximum
- (c) If $A(v_{max}) < 0$, $S_1 = S_1 \cup v$
- (d) else if $A(v_{max}) > 0$, $S_2 = S_2 \cup v$
- (e) else add v uniformly at random to S_1 or S_2

4. **repeat**

- (a) OldCut = $w_H(S_1)$
- (b) If $v \in S_1$ and $A(v) > 0$, move v from S_1 to S_2
- (c) If $v \in S_2$ and $A(v) < 0$, move v from S_2 to S_1

until $w_H(S_1) \leq \text{OldCut}$

- 5. If $w_H(S_1) > \text{BestCut}$, $\text{BestCut} = w_H(S_1)$

Final: Return BestCut

The first phase of the above algorithm (Step 1-3) is designed to find a cut in which the highly negative weight edges do not cross the cut. The cut is initialized using an edge of the graph and the algorithm is repeated enough times to make sure that every edge is considered. Step 4 by itself is exactly the well-known Greedy algorithm for computing max-cuts (Sahni and Gonzalez, 1974).

6.3 Results

We used the filtered HuRef data from Levy et al., 2007 to evaluate our algorithm HapCUT. The data contains a total of 1.85 million heterozygous variants for the 22 autosomes. As a typical example, chromosome 22 contained 24,968 variants encoded by 103,356 fragments. Only 53,279 of these cover more than one variant and are therefore useful for haplotype assembly. 18,119 of these fragments correspond to mate-pairs. The chromosome is partitioned into 609 dis-connected variant “blocks” or connected component based on the links between variants in addition to 921 isolated

variants. These blocks provide large haplotypes, clearly illustrating the power of this haplotype assembly. However, as the length of our haplotypes is predetermined by the connected components, we do not discuss this further, referring the interested reader to Levy et al., 2007. The haplotypes for each of these blocks are assembled independently. The average number of calls for a variant is 6.7 (see Figure 6.1(a) for distribution of the number of variant calls per fragment). A fragment spans 9.67 variants but has only 3.25 variant calls on the average (see Figure 6.1(b) for distribution of the difference between span and length). This clearly indicates the highly “gapped” nature of the fragment data.

6.3.1 MEC scores for HuRef chromosomes

We ran HapCUT for each of the HuRef chromosomes. For each chromosome, the algorithm was initialized with a randomly chosen haplotype. We found that the MEC score improves as we iteratively compute cuts and change the haplotypes. Most of the improvement in the MEC score happens in the first few iterations with no further improvement after 40-50 iterations. We compared the MEC scores for the HuRef data using four different algorithms: i) the greedy heuristic of Levy et al., 2007, ii) Fast Hare (implemented as described in Panconesi and Sozio (2004)), iii) the Markov chain Monte Carlo algorithm HASH and iv) our algorithm HapCUT (see Figure 6.2).

HapCUT performs significantly better than the greedy heuristic (the MEC scores are 20-25% lower for all chromosomes) and very similar to HASH. The performance of the heuristic Fast Hare is generally worse than that of the greedy heuristic. For a few connected components, the MEC score for HASH was slightly lower than that of HapCUT (run for 100 iterations). On all of these cases, a greedy choice of the cut did not improve the MEC score (data not shown). Clearly, an MCMC sampling approach has the advantage of being able to make sub-optimal choices and thereby reach sample a slightly better haplotype. On the other hand, for some of the chromosomes, we observed that the total MEC score of HapCUT was better than that of HASH. A possible explanation is that while the objective of HapCUT is to find the optimal MEC solution,

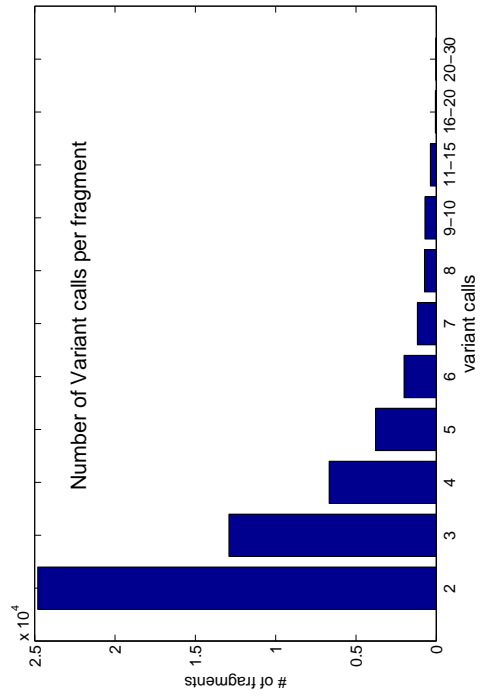
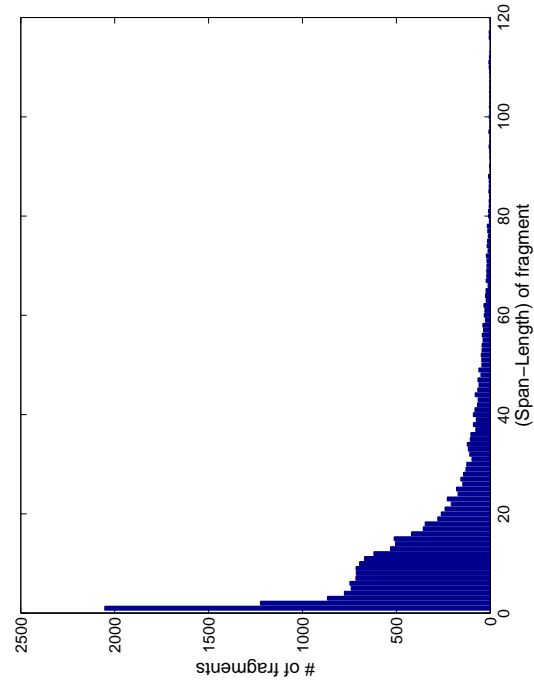


Figure 6.1: (a) Distribution of number of variant calls per fragment. Fragments that cover only one variant site (about half of the total fragments) are not shown. (b) Distribution of the difference between the “span” of a fragment (difference between the last variant call and the first variant call) and the “length” (# of variant calls). Fragments with span equal to length (35159/53278 fragments) are not included to improve clarity. Both plots are for chromosome 22 from HuRef genome.

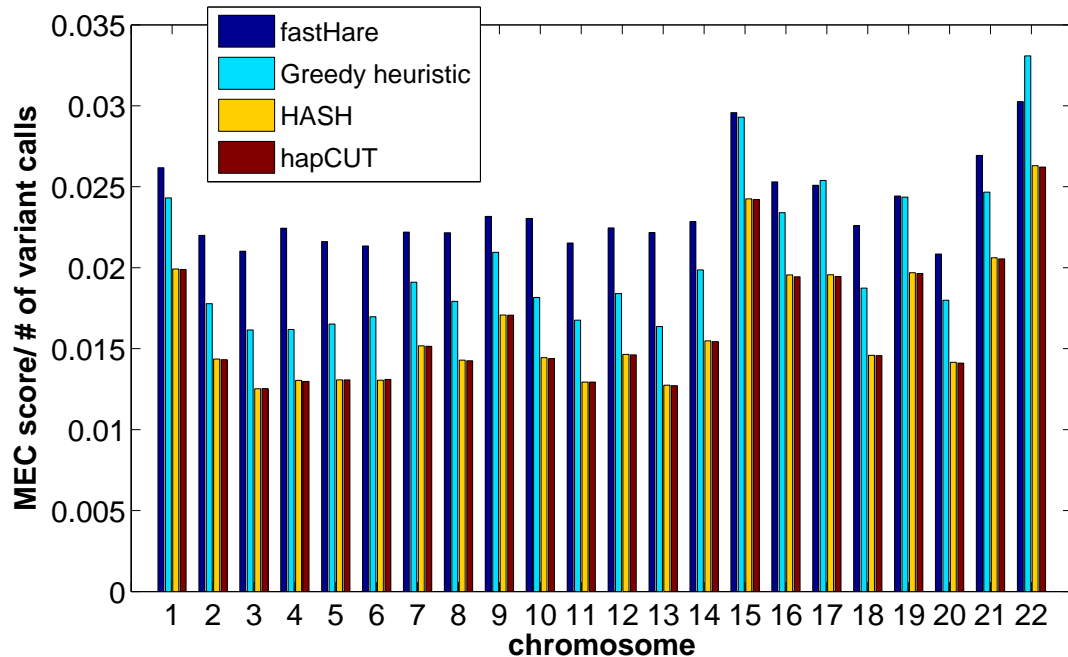


Figure 6.2: MEC scores (divided by the number of variant calls) for the HuRef chromosomes for four different methods. The performance of HapCUT and HASH is comparable, and significantly better than the greedy heuristic and Fast Hare.

HASH is geared towards finding the maximum likelihood solution. The two may be different when fragments have lengths greater than 2. HapCUT also offers the advantage of fast computation time in comparison to MCMC sampling algorithms such as HASH. For chromosome 22, HapCUT takes less than 30 minutes to compute the MEC score while HASH takes more than 10 hours. For all chromosomes, HapCUT was an order of magnitude faster than HASH (results not shown).

6.3.2 Simulations using HuRef data

We tested the performance of HapCUT on simulated data generated using the HuRef chromosomes. The fragment matrix for a chromosome was suitably modified to make it “error free” or perfectly consistent with a particular haplotype. To generate a fragment matrix with error rate of ε ($0 \leq \varepsilon \leq 0.5$), each variant call in the fragment matrix was flipped with probability ε . We ran HapCUT on this modified fragment matrix

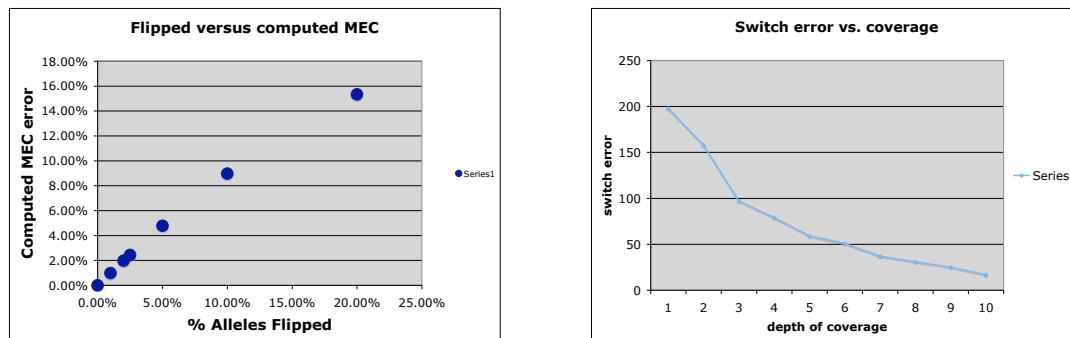


Figure 6.3: (a): Number of Simulated versus estimated errors and (b) Haplotype switch error as a function of depth of coverage (number of calls for a variant), for $\varepsilon = 0.02$. The switch error decreases with increasing depth.

and compared the reconstructed haplotypes with the true haplotypes. In Figure 6.3(a), we plot the best MEC score (scaled by the number of variant calls) against the simulated error rate, i.e. the fraction of variant calls that were flipped. We observe that the MEC score is always less than the number of flipped calls and ratio of the MEC score to the number of flipped variant calls decreases as the error rate increases. This is to be expected, because as the number of flipped variant calls increases, some calls might become consistent with a different (lower MEC) haplotype.

Flipped base-calls could also result in errors in the reconstructed haplotypes. If the depth of coverage is low, very little can be done to recover from the error. Also, if ε is high, the optimal haplotype could well be different from the one we started with. However, we expect that at high depths of coverage and low-error rates, a correct haplotype can be recovered accurately. In Figure 6.3(b) we plot the switch error (number of switches between the original haplotype and the reconstructed haplotype) against the depth of coverage, i.e. for a particular value on the x-axis, we ignore variants with coverage below that value for computing the switch error. One can clearly see that as the depth increases, the switch error decreases from 198 (1.11% of the sites) to 17 (0.098%).

Not surprisingly, we also find that the flipped variants are largely the same as the one that mismatch the computed haplotype. Of the 3321 fragments for which some variant call was flipped, we identified 3213 that also mismatched against the computed

haplotype. An additional 146 fragments that did not contain flipped alleles mismatched with the computed haplotype. Overall, these results indicate the robustness of our solution to errors in the data.

6.4 Maximum Likelihood estimate of the Switch error rate of HuRef haplotypes

The MEC score measures the consistency of the haplotype assembly with the fragment matrix. However, we are also interested in the absolute accuracy of the inferred haplotypes. The absolute accuracy can be measured using the switch error rate, which is the fraction of adjacent pairs of sites in the HuRef individual whose phase is incorrect. We have used the phased haplotypes from the HapMap project (The International HapMap Consortium, 2005) to obtain a Maximum Likelihood estimate of the absolute accuracy of the HuRef haplotypes. As the Huref individual is European in origin, we compare the HuRef haplotypes H against the set of 120 CEU HapMap haplotypes restricted to the set of SNPs heterozygous in the HuRef individual. The two HuRef haplotypes are a mosaic of the population haplotypes and a direct comparison of the full haplotypes with the HapMap haplotypes is not possible. We compute the likelihood of the haplotype H conditional on the CEU haplotypes and as a function of the switch error rate.

Consider a pair of adjacent SNPs i, j heterozygous in the HuRef individual. Let f_{00}, f_{01}, f_{10} and f_{11} be the frequencies of the four pairs in the HapMap CEU sample. If H_t denote the true HuRef haplotypes (unobserved), the likelihood of the phasing $H_t[i, j]$ being (00, 11) is given by

$$Pr(H_t[i, j] = (00, 11)) = \frac{f_{00}f_{11}}{f_{00}f_{11} + f_{01}f_{10}}$$

We can similarly compute the probability $Pr(H_t[i, j] = (01, 10))$. For a *switch error rate* ε_s , the likelihood of the phasing between the pair (i, j) is given by

$$L_H(i, j) = (1 - \varepsilon_s)Pr(H_t[i, j] = H[i, j]) + \varepsilon_s Pr(H_t[i, j] \neq H[i, j])$$

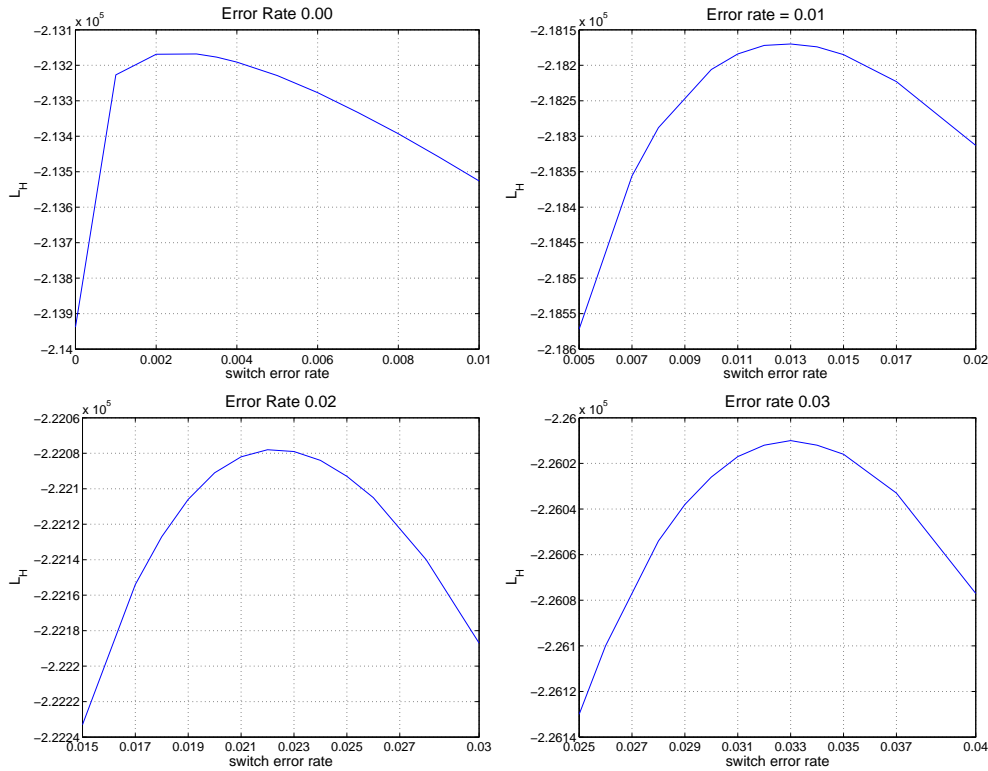


Figure 6.4: Haplotype log-likelihood curves for four different values of the switch error rate.

The switch errors between H and H_t are a result of sequencing errors and likely to be distributed independent of LD. Therefore we can assume the switch error rate ε_s to be uniform for all pairs. We approximate the likelihood of the full haplotype H as

$$L_H = \prod_{i=1}^{n-1} L_H(i, i+1)$$

The haplotype likelihood (L_H) is a function of the switch error rate ε_s and the LD in the HapMap haplotypes. If there are very few switch errors between H and H_t , then the likelihood function L_H is expected to be maximum for values of ε_s close to 0. As the number of switch errors increases, the contribution of the second term in the likelihood $L_H(i, j)$ increases and therefore the Maximum Likelihood (ML) estimate of ε_s should increase. We have used the HapMap haplotypes to evaluate if the Maximum

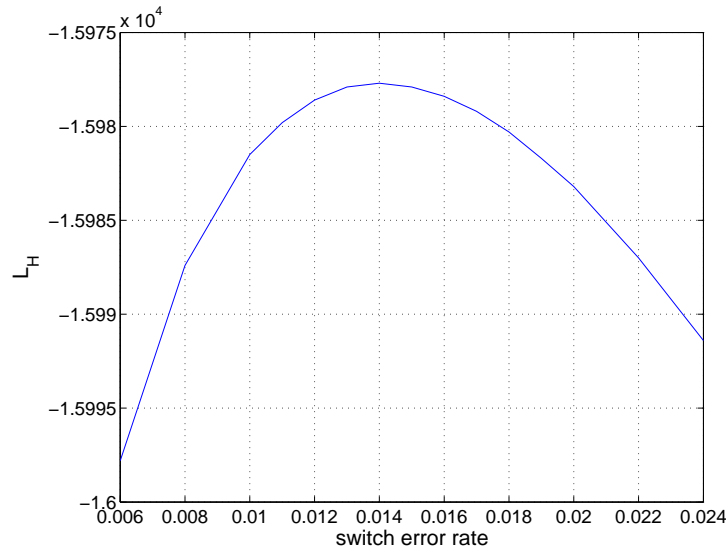


Figure 6.5: The log-likelihood curve for the HuRef haplotypes for chromosome 22.

Likelihood estimator is a good estimate of the switch error rate. The haplotypes for one of the 60 CEU individuals was chosen to represent the true haplotypes H_t . We then simulated switch errors randomly with varying switch error rates ε_s (0 to 0.05) to generate the haplotype pair H . The likelihood function L_H was then plotted for different values of the error rate (see Figure 6.4 for likelihood curves for four different values of ε_s). From the likelihood curves, we see that the ML estimate is a good estimate of the switch error rate with a tendency to slightly over-estimate the switch error rate. For a simulated switch error rate of 0.01, the likelihood was maximum for $\varepsilon_s = 0.013$. Similarly for an error rate of 0.02, the ML estimate was 0.022.

We then plotted the haplotype likelihood (L_H) for the HuRef haplotypes (inferred using HapCUT) as a function of the switch error rate ε_s (see Figure 6 for a plot for chromosome 22). The likelihood curve is flat in the region 0.013-0.015 with a maxima at 0.014. From this, we can infer that the switch error rate of the HuRef haplotypes is slightly more than 1% but no more than 1.4%. In comparison, the switch error rate of haplotypes inferred from CEU population data is 5.4% for unrelated individuals and 0.53% for parent-child trios (Marchini et al., 2006).

6.5 Acknowledgement

Chapter 6, in full, will appear in the Proceedings of the European Conference on Bioinformatics, 2008, V. Bansal and V. Bafna, “HapCUT: an Efficient and Accurate Algorithm for the Haplotype Assembly Problem”. The dissertation author was the primary investigator and author of this paper.

Chapter 7

Markov chains for Haplotype

Assembly: Mixing time analysis

7.1 Introduction

In chapter 5, we proposed a Markov chain Monte Carlo framework for the haplotype assembly problem. Within this MCMC framework, we saw how different collection of subsets of the fragment matrix result in different Markov chains. In this chapter, we analyze the mixing times of two Markov chains for a family of fragment matrices. We derive explicit lower and upper bounds on the convergence of the Markov chains that demonstrate that a cut-based Markov chain is more efficient at sampling the haplotype space. The upper and lower bounds on convergence times are derived using novel coupling, and conductance based techniques (Sinclair and Jerrum, 1989; Jerrum and Sinclair, 1997). These results provide theoretical justification for the better performance of our cut-based MCMC algorithm, HASH, for haplotype assembly and also are interesting in their own right. Before we present these results, we give some formal definitions about certain properties of Markov chains and the concept of mixing time.

Definition 7: Let \mathcal{M} be a Markov chain on a finite state space Ω with transition matrix $P = (p_{ij})_{i,j \in \Omega}$. The chain is said to be ergodic if it is irreducible, i.e. $P_{ij}^n > 0$ for

some natural number $n > 0$ and all $i, j \in \Omega$ and aperiodic. A Markov chain is said to be *time-reversible* if there exists a probability distribution $\pi = (\pi_i)_{i \in \Omega}$ that satisfies the *detailed balance* condition

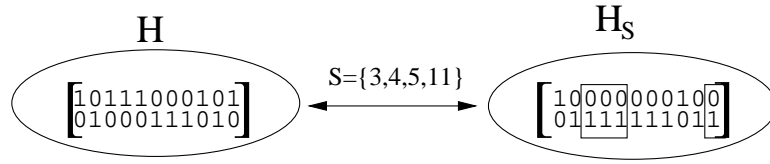
$$p_{ij}\pi_i = p_{ji}\pi_j = Q(i, j), \text{ for all } i, j \in \Omega$$

It is well-known that this π satisfying the detailed-balance condition is the unique stationary distribution, as

$$(\pi P)[j] = \sum_i \pi_i P_{ij} = \sum_i \pi_j P_{ji} = \pi_j$$

Within our Markov chain Monte Carlo framework, a fragment matrix X with n columns, a corresponding matrix q of error probabilities and a collection Γ of subsets of the columns of X defined a Markov chain denoted by $\mathcal{M}(X, q, \Gamma)$. A fragment matrix is a matrix X with m rows where each row is a string of length n over the alphabet $\{0, 1, -\}$. Corresponding to this fragment matrix, we have a matrix q of error probabilities where $0 \leq q_i[j] \leq 1$ if $X_i[j] \neq -$ and $q_i[j] = -$ if $X_i[j] = -$. For brevity, we will omit X and q whenever they are implicit, and denote the Markov chain as simply $\mathcal{M}(\Gamma)$. A state of this Markov chain is an unordered haplotype pair $H = (h_1, h_2)$ where h_1 and h_2 are binary strings of length n over the alphabet $\{0, 1\}$. Moreover, h_2 is the bit-wise complement of h_1 . The state space is the set of 2^{n-1} haplotype pairs. The transition probabilities are defined based on the collection of subsets Γ . For every state H , there are $k + 1$ possible transitions, including the self-loop, where k is the number of subsets in Γ .

Every transition of the Markov chain is of the form: $H \rightarrow H_S$ where H is the current state (haplotype pair) and S is a subset of columns of X . For example,



The probability of the transition $H \rightarrow H_S$ is $\frac{1}{2k} \min \left[1, \frac{Pr(X|H_S^t, q)}{Pr(X|H^t, q)} \right]$ and the probability of the self-loop $H \rightarrow H$ is $\frac{1}{2}$. These probabilities completely define the transition matrix of the Markov chain $\mathcal{M}(\Gamma)$. Our first result is to show that $\mathcal{M}(\Gamma)$ has stationary distribution $Pr(X|H, q) (\approx Pr(H|X, q))$, provided Γ includes a minimal collection of subsets. Formally,

Theorem 21: Let $\Gamma_1 = \{\{1\}, \{2\}, \dots, \{n\}\}$. For every fragment matrix X , error probabilities $q_i[j] > 0 (\forall i, j)$, and any $\Gamma \supseteq \Gamma_1$, $\mathcal{M}(\Gamma)$ is *ergodic* (i.e. *irreducible & aperiodic*) with $Pr(X|H, q)$ as its stationary distribution.

Proof: If $\Gamma_1 \subseteq \Gamma$, then starting from any haplotype configuration H_1 we can reach any other haplotype configuration H_2 by flipping the locations (in an arbitrary order) that they differ in. The probability of this sequence of transitions is non-zero since $q_i[j] > 0 \forall i, j$. Hence, for any pair of haplotypes H_1 and H_2 , $P^t(H_1, H_2) > 0$ for some $t < n$ where n is the number of columns. Aperiodicity is ensured by the fact that with probability at least $1/2$, the Markov chain remains in the state that it is currently in. Finally, it can be verified that the detailed balance condition is satisfied for $\pi_H = Pr(X|H, q)$, as for any neighboring H, H' ,

$$\pi_H Pr(H, H') = \pi_{H'} Pr(H', H) = \min\{Pr(X|H, q), Pr(X|H', q)\}$$

Therefore, the stationary distribution of the chain is $Pr(X|H, q)$.



7.2 Mixing time of Markov chains

Mixing time represents the number of steps after which the Markov chain is guaranteed to be “almost” sampling from the required posterior distribution. From the perspective of sampling using Markov chains, mixing time is similar to the running time of an algorithm, i.e. a low mixing time corresponds to an efficient Markov chain. More formally, mixing time of a Markov chain is defined based on the distance of the chain from the stationary distribution.

Definition 8: The distance of the Markov chain from the stationary distribution π at time t is defined as

$$\|P^t, \pi\| = \max_{i \in \Omega} \frac{1}{2} \sum_{j \in \Omega} |P_{ij}^t - \pi_j|$$

For any $\epsilon > 0$, the mixing time of the Markov chain is defined as

$$\tau(\epsilon) = \min\{t : \|P^{t'}, \pi\| \leq \epsilon, \forall t' \geq t\}$$

A Markov chain is said to be rapidly mixing if the mixing time of chain can be bounded from above by a polynomial in n and $\log \epsilon^{-1}$ where n is the size of each state of the chain. Several techniques have been developed for analyzing the mixing time of a Markov chain in the computer science community (see Jerrum and Sinclair (1997) for an excellent review of these methods). In the remainder of this section, we describe some of these techniques and results that we will subsequently use for our analysis.

7.2.1 Conductance and Mixing time

Definition 9: For a finite spaced Markov chain \mathcal{M} , consider the undirected weighted graph with vertex set Ω and edge set $E = \{(x, y) \in \Omega^2 : Q(x, y) > 0\}$. The conductance (Sinclair and Jerrum, 1989) of the Markov chain \mathcal{M} is defined as

$$\Phi(\mathcal{M}) = \min_{S \subset \Omega, 0 < \pi(S) < 1/2} \left(\frac{Q(S, \bar{S})}{\pi(S)} \right)$$

where $Q(S, \bar{S}) = \sum Q(x, y)$ for all pairs (x, y) such that $x \in S$ and $y \in \bar{S}$.

Intuitively, the conductance measures the ability of the Markov chain to escape from any subset of the state space. A low conductance implies that the conditional probability of the Markov chain escaping from some subset $\mathcal{S} \in \Omega$ in a single step is small. The following result tightly relates the mixing time of a reversible Markov chain to the conductance.

Theorem 22: [(Sinclair and Jerrum, 1989; Sinclair, 1992; Randall, 2006)] Let \mathcal{M} be a finite, reversible, ergodic Markov chain with loop probabilities $P(x, x) \geq 1/2$ for all states x . Then

$$\frac{\log((2\epsilon)^{-1})}{4\Phi} \leq \tau(\epsilon) \leq \frac{2 \log((\pi_y \epsilon)^{-1})}{\Phi^2}$$

for any choice of initial state y .

The mixing time of a Markov chain can also be related to the spectral gap of the transition matrix P . Let $Gap(P) = \lambda_0 - |\lambda_1|$ be the spectral gap of P where $\lambda_0, \lambda_1, \dots, \lambda_{|\Omega|-1}$ are the eigenvalues of P . Also, $1 = \lambda_0 > |\lambda_1| \geq \lambda_i$ for all $i \geq 2$. If $P(x, x) \geq 1/2$ for all $x \in \Omega$, the spectral gap tightly bounds the mixing time on both sides:

Theorem 23: [Sinclair (1992)]

$$\frac{1}{2Gap(P)} \log \left(\frac{1}{2\epsilon} \right) \leq \tau(\epsilon) \leq \frac{1}{Gap(P)} \log \left(\frac{1}{(\min_{x \in \Omega} \pi_x) \epsilon} \right)$$

7.2.2 The Coupling Argument

A coupling for a Markov chain \mathcal{M} is a stochastic process (X_t, Y_t) on $\Omega \times \Omega$ such that (X_t) (or (Y_t)) considered marginally is identical to the Markov chain \mathcal{M} . The Coupling Lemma (Aldous, 1986) states that the distance of a Markov chain \mathcal{M} from its stationary distribution π is bounded from above by $Pr[X_t \neq Y_t]$, i.e. the probability that the coupling (X_t, Y_t) for \mathcal{M} has not coupled. This holds for any coupling, therefore, if one can construct a coupling with low coupling time, it implies that the corresponding Markov chain has low mixing time. We present a formal definition of a coupling:

Definition 10: A coupling for a Markov chain \mathcal{M} is a stochastic process (X_t, Y_t) on $\Omega \times \Omega$ with the following two properties:

1. $Pr[X_{t+1} = y | X_t = x] = P(x, y) = Pr[Y_{t+1} = y | Y_t = x]$
2. if $X_t = Y_t$, then $X_{t+1} = Y_{t+1}$

We will use the following theorem:

Theorem 24: Let (X_t, Y_t) be any coupling for the Markov chain \mathcal{M} and let δ be any integer valued metric defined on $\Omega \times \Omega$. Suppose that there exists a positive $\beta \leq 1$ such that

$$\mathbf{E}[\delta(X_{t+1}, Y_{t+1})] \leq \beta \delta(X_t, Y_t)$$

for all t . Let D be the maximum value of the metric δ . For $\beta < 1$, the mixing time $\tau(\epsilon)$ of the Markov chain \mathcal{M} satisfies

$$\tau(\epsilon) \leq \frac{\log(D\epsilon^{-1})}{(1 - \beta)}$$

For $\beta = 1$, if there exists an $\alpha > 0$ such that

$$Pr[\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)] \geq \alpha$$

for all t , then the mixing time $\tau(\epsilon)$ is bounded as

$$\tau(\epsilon) \leq \lceil \frac{eD^2}{\alpha} \rceil \lceil \log(\epsilon^{-1}) \rceil$$

7.2.3 The canonical paths argument

Another technique for proving upper bounds on the mixing time of a Markov chain is what is called the canonical path argument. Consider the undirected weighted graph underlying the Markov chain \mathcal{M} as defined previously. For each ordered pair of states $(x, y) \in \Omega^2$, define a canonical path ψ_{xy} that goes from x to y using edges of

the graph. Let Ψ be the set of all canonical paths. Define the congestion of the set of canonical paths Ψ as

$$\rho(\Psi) = \max_{e \in E} \frac{1}{Q(e)} \sum_{\psi_{xy}, e \in \psi_{xy}} \pi(x)\pi(y)|\psi_{xy}|$$

where $|\psi_{xy}|$ is the length of the canonical path ψ_{xy} . Intuitively, a Markov chain will have a low mixing time if no edge is heavily loaded, i.e. there exists a set of canonical paths Ψ for which $\rho(\Psi)$ is not large. The following results formalizes this notion.

Theorem 25: [Sinclair (1992)] Let \mathcal{M} be a finite, reversible, ergodic Markov chain with loop probabilities $P(x, x) \geq 1/2$ for all states x . For any set of canonical paths Ψ , the mixing time of \mathcal{M} satisfies

$$\tau(\epsilon) \leq \rho(\Psi) \left(\ln \left(\frac{1}{\pi(y)\epsilon} \right) \right)$$

for any choice of initial state y .

The canonical path argument represents an useful algorithmic technique for analyzing the mixing time of Markov chains.

7.2.4 Decomposition Theorem for analyzing mixing time

For many Markov chains, it is not easy to directly analyze the mixing time using one of the techniques described previously. Madras and Randall (2002) introduced a new approach to analyze the mixing time of a Markov chain based on decomposing the state space of a Markov chain into smaller pieces, each of which can then be analyzed using the standard techniques. Martin and Randall (2006) presented similar results for a disjoint decomposition of the state space. Informally, these decomposition results show that if the state space of a Markov chain can be decomposed into disjoint subsets such that the Markov chain restricted to each of these subsets is rapidly mixing and, in

addition, there is sufficient ergodic flow between the subsets, the original Markov chain is rapidly mixing.

Let $\Omega_1, \Omega_2, \dots, \Omega_m$ be disjoint subsets of Ω such that $\cup_i \Omega_i = \Omega$. We define the Markov chain restriction \mathcal{M}_i of \mathcal{M} to Ω_i as the Markov chain obtained by rejecting all moves between elements of Ω_i and those outside this subset. More formally, if $x \neq y$ and $x, y \in \Omega_i$, then $P_{\Omega_i}(x, y) = P(x, y)$ and $P_{\Omega_i}(x, x) = 1 - \sum_{y \in \Omega_i, y \neq x} P_{\Omega_i}(x, y)$.

The Markov chain that moves between the subsets $\Omega_1, \Omega_2, \dots, \Omega_m$ is called the projection Markov chain \mathcal{M}_H and is defined on the set $[m] = \{1, 2, \dots, m\}$ where each point i corresponds to the set Ω_i . The transition matrix $P_H(i, j)$ is defined as

$$P_H(i, j) = \frac{1}{\pi(\Omega_i)} \sum_{x \in \Omega_i, y \in \Omega_j} \pi(x) P(x, y)$$

Note that the Markov chain \mathcal{M}_H is also a reversible and ergodic Markov chain on $[m]$.

Theorem 26: [Martin and Randall (2006)] Let P_{Ω_i} and P_M be as defined above. Then

$$Gap(P) \geq \frac{1}{2} Gap(P_H) \min_{i=1 \dots m} Gap(P_{\Omega_i})$$

Using this inequality, the mixing time of the chain \mathcal{M} can be bounded using upper bounds on the mixing time of \mathcal{M}_H and each of the chains \mathcal{M}_i .

7.3 A Family of Fragment Matrices

We will analyze the mixing time of Markov chains for a family $\mathcal{X}_{d,n}$ of matrices. Figure 7.1 depicts an example $X \in \mathcal{X}_{2,n}$. For any $X \in \mathcal{X}_{d,n}$, there are a total of n sites, and $d \cdot (n - 2) + 2$ fragments. Each fragment occupies exactly 2 adjacent sites (columns) in X . For each adjacent pair of positions $(i, i + 1)$ except the middle $(i = \lfloor n/2 \rfloor)$, there are exactly d fragments, each supporting the phasing $(00, 11)$. At $i = \lfloor n/2 \rfloor$, we have two fragments 00 , and 01 , which support either phasing. There are exactly two optimal

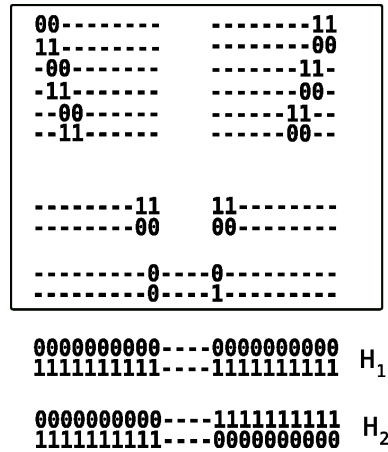


Figure 7.1: A fragment matrix $\mathcal{X}_{d,n}$ ($n = 20, d = 2$ as shown) for which two haplotypes H_1 and H_2 have equal likelihood.

haplotypes

$$H^1 = \begin{pmatrix} 00000 \dots 00000 \\ 11111 \dots 11111 \end{pmatrix} \qquad H^2 = \begin{pmatrix} 00000 \dots 11111 \\ 11111 \dots 00000 \end{pmatrix}$$

Consider the haplotype H as two paths (corresponding to h, \bar{h}) from position 1 to n . Using this, we can *switch* paths to move from one haplotype to another. For example, we need 'one switch' to move from H^1 to H^2 . Let $H \oplus H'$ denote the *switch-distance* (the number of switches needed to transform one into the other) between haplotypes H , and H' . For any haplotype H , we define $D(H) = \min\{H \oplus H^1, H \oplus H^2\}$. Note that, $D(\cdot)$ is a function from the state space Ω of the Markov chain to the set of integers $\{0, 1 \dots, n - 2\}$. For $0 \leq k \leq (n - 2)$, define

$$\mathcal{D}^k = \{H \mid \min\{H \oplus H^1, H \oplus H^2\} = k\}$$

It follows that $\mathcal{D}^0 = \{H^1, H^2\}$ and for all $H \in \mathcal{D}^k$,

$$\pi(H) = \rho^k \pi(H^1) = \rho^k \pi(H^2)$$

The cardinality of the set \mathcal{D}^k is $2 \binom{n-2}{k}$.

Lemma 27:

$$|\mathcal{D}^k| = 2 \binom{n-2}{k}$$

Proof: Once the k positions where the haplotype switches from 0 to 1 or 1 to 0 are chosen, a unique haplotype is defined. Using this definition, the total number of haplotypes pairs is $2 \sum_k \binom{n-2}{k} = 2^{n-1}$.



Next, we show that $\pi(H^1)$ and $\pi(H^2)$ dominate the probability space for small enough values of ρ .

Lemma 28: For $\rho < \frac{1}{2n}$, $\pi(H^1) + \pi(H^2) > \frac{1}{2}$.

Proof:

$$\begin{aligned} 1 &= \sum_H \pi(H) \\ &= \sum_{k=0}^{n-2} 2 \binom{n-2}{k} \rho^k \pi(H^1) \\ &= 2\pi(H^1) (1 + \rho)^{n-2} \\ \rightarrow (\pi(H^1) + \pi(H^2)) &= (1 + \rho)^{-(n-2)} \\ (\pi(H^1) + \pi(H^2)) &= \left((1 + \rho)^{\frac{1}{\rho} + 1} \right)^{-(n-2) \frac{\rho}{1+\rho}} \end{aligned}$$

Using the inequality $(1 + \frac{1}{k})^{k+1} > e$ (for all $k > 1$),

$$(\pi(H^1) + \pi(H^2)) \geq (e)^{-\frac{\rho}{1+\rho}(n-2)}$$

For $\rho \leq \frac{1}{2n}$, we have

$$(\pi(H^1) + \pi(H^2)) \geq \frac{1}{\sqrt{e}} > 1/2$$



7.3.1 $\mathcal{M}(\Gamma_1)$ has poor mixing time

We use the conductance argument to show that the mixing time for the Markov chain $\mathcal{M}(X_{d,n}, q, \Gamma_1)$ grows exponentially with d , the depth of fragment coverage.

Theorem 29: Consider a fragment matrix $X(n, d)$ with n columns and $d * (n - 2) + 2$ rows with structure as described in Figure 7.1. Let q be the uniform error probability for all positions and define $\rho = \left(\frac{2pq}{p^2+q^2}\right)^d$. For any $\epsilon > 0$, the mixing time of the Markov chain $\mathcal{M}(\Gamma_1, X(n, d), q)$ is $\Omega\left(\log\left(\frac{1}{\epsilon}\right) n \rho^{-1}\right)$.

Proof: Choose $S = \{H^1\}$. There are exactly n edges between S and \bar{S} . Of these $n - 4$ have load exactly $\rho^2 \pi(H^1)$ and 4 edges $(1, n/2, n/2 + 1, n)$ have weight $\rho \pi(H^1)$. Therefore,

$$Q(S, \bar{S}) = \frac{n-4}{n} \rho^2 \pi(H^1) + \frac{4}{n} \rho \pi(H^1) \leq \left(\rho^2 + \frac{4\rho}{n}\right) \pi(H^1)$$

and

$$\pi(S) = \pi(H^1)$$

Therefore $\phi \leq \rho^2 + \frac{4\rho}{n} \leq 5\frac{\rho}{n}$ and the bound on the mixing time follows.



7.4 A Markov chain with polynomial mixing time

For the fragment matrix $X_{d,n}$ shown in Figure 7.1, if we include the subset $S_{1\dots \frac{n}{2}}$ (columns 1 to $n/2$) in Γ , the Markov chain in state H_1 will move to H_2 whenever this subset is sampled and vice versa. Intuition suggests that the mixing time of the Markov chain should also decrease since the bottleneck in moving between H_1 and H_2 has been removed. However, proving that the mixing time of the Markov chain $\mathcal{M}(\Gamma_1 \cup S_{1\dots n/2})$ does not grow exponentially with d is not so easy. Using a combination of tools for analyzing the mixing time of Markov chains, we will demonstrate that the mixing time of the Markov chain resulting from $\Gamma = \Gamma_1 \cup S_{1\dots n/2}$ is polynomial in n (the number of columns) and d . The following theorem represents the main result of this chapter.

Theorem 30: Let $X \in \mathcal{X}_{d,n}$ be chosen arbitrarily, and $q > 0$ be the uniform error probability for all positions. For any $\epsilon > 0$, the mixing time for the Markov chain $\mathcal{M}(\Gamma_1 + S_{1\dots n/2}, X_{d,n}, q)$ is $O(n^{10} \ln(\epsilon^{-1}))$.

We will use the Markov chain decomposition tool (MARTIN and RANDALL, 2006) (see Theorem 26) to prove this result. Our decomposition of the state space is very simple: every $D^k, 0 \leq k \leq n - 2$ is a subset corresponds to a subset in the decomposition. In other words, there are $n - 1$ subsets. Note that, the set D^k has $2^{\binom{n-2}{k}}$ states/haplotypes all of which have the same probability $\pi(H)$. We denote the Markov chain on the set of haplotype D^k by \mathcal{M}_k . Using the coupling technique, we will prove that the mixing time of each of the chains \mathcal{M}_k is bounded by a polynomial in n . We will use the canonical paths method to bound the mixing time of the projection Markov chain \mathcal{M}_H . Using these bounds and Theorem 11, we will prove a bound on the mixing time of the full Markov chain.

7.4.1 Mixing time of the Markov chain restricted to D^k

Theorem 31: Let \mathcal{M}_k denote the Markov chain $\mathcal{M}(\Gamma_1 + S_{1\dots n/2}, X_{d,n}, q)$ restricted to the set D^k . Then, the mixing time of \mathcal{M}_k is bounded as follows:

$$\tau(\epsilon) \leq c(n+1)^3 k^2 \lceil \log(\epsilon^{-1}) \rceil$$

where c is a small constant.

Proof: We will construct a coupling (X_t, Y_t) for this Markov chain. Recall that the set D^k represents all haplotypes whose minimum switch distance from H^1 or H^2 is k . Therefore, any haplotype in D^k can be represented as an integer vector of length k : (x_1, x_2, \dots, x_k) which represent the positions (*frontiers*) where the haplotype switches from 0 to 1 or 1 to 0, except for the position $X_{\frac{n}{2}+1}$. In order to include a possibility of a switch in the middle (this does not affect the probability of the haplotype), we include an extra bit for each haplotype. Therefore, every haplotype in D^k is represented by a length

k integer vector padded with a single bit $((x_1, x_2, \dots, x_k), b)$ where $0 < x_1 < x_2 \dots < x_k < n$. The “middle region” from x_i to x_{i+1} , where $x_i \leq n/2 \leq x_{i+1}$ is always assumed to be a single block even when there is an extra-flip at $x_{\frac{n}{2}+1}$. In the best case, there are $2k + 1$ new haplotypes that can be formed from a haplotype in D^k (each of the k frontiers can be moved to the left and right, and additionally, each haplotype can be flipped in the middle by setting $b \in \{0, 1\}$). The transition probability of each of these moves is $\frac{1}{2^{(n+1)}}$. With all the remaining probability, the Markov chain \mathcal{M}_k (restriction of \mathcal{M} to D^k) stays in the current haplotype. We denote the haplotype formed by moving the i -th frontier to the left as $X(i^-) = ((x_1, \dots, x_i^-, \dots, x_k), b)$ and the one formed by moving it to the right as $X(i^+) = ((x_1, \dots, x_i^+, \dots, x_k), b)$. Note that if $x_{i+1} = x_i + 1$, we cannot move the frontier to the right. Similarly, if $x_{i-1} = x_i - 1$, we cannot move the frontier left. Another observation that will be useful is the following. Consider the block (x_i, x_{i+1}) . If the length of this block is 2 or more, we can flip the bit x_i to transform the haplotype into $((x_1, x_i + 1, x_{i+1}, \dots, x_k), b) \in D^k$. Similarly, we can flip the bit $x_{i+1} - 1$ to transform the haplotype into $((x_1, x_i, x_{i+1} - 1, \dots, x_k), b) \in D^k$. In either case, the frontier position shifts right or left by one. The first block is represented as $(x_0 = 1, x_1 - 1)$ and the last block as (x_k, n) .

Let $X = ((x_1, x_2, \dots, x_k), b_x)$ and $Y = ((y_1, y_2, \dots, y_k), b_y)$ be any two haplotypes in D^k . We define a distance metric δ between these two haplotypes as

$$\delta(X, Y) = \left(\sum_{i=1 \dots k} |x_i - y_i| \right) + |b_x - b_y|$$

This distance metric δ is what we will use for constructing the coupling. Note that if $\delta(X, Y) = 0$, the two haplotypes are identical. The following lemma proves that the above distance metric is a valid metric in the sense that there is a sequence of valid transitions of length $\delta(X, Y)$ that transforms the haplotype X into Y . Also, the maximum value that this metric achieves on D^k is bounded by $(n + 1)k$.

Lemma 32: For any pair of haplotypes (X_t, Y_t) where $X_t, Y_t \in D^k$, there exists a sequence of transitions in \mathcal{M}_k of length $\delta(X_t, Y_t)$.

Proof: We use induction on k , the number of frontiers to prove the result. The result clearly holds for $k = 0$. There are only two haplotypes in D^0 and the distance between these two haplotypes is 1. Flipping the subset $S_{1\dots n/2}$ transforms one haplotype into the other. Note that in general, if $b_x \neq b_y$, the first transition will be to flip the subset $S_{1\dots n/2}$ of one haplotype to make b_x and b_y equal.

Let us assume that the result holds for $k = r$, i.e. for any two haplotypes of length $\leq n$ and r frontiers, there is a sequence of transitions with length $\delta(X_t, Y_t)$ that transforms one haplotype into another. Let X and Y be two haplotypes in D^{r+1} . Consider the first frontier of X and Y starting from the left. If $x_1 = y_1$, then we can ignore the part of X and Y before the first frontier and consider the two haplotypes $X_{-1} = (x_2, \dots, x_{r+1})$ and $Y_{-1} = (y_2, \dots, y_{r+1})$ each with r frontiers. Clearly, $\delta(X, Y) = 0 + \delta(X_{-1}, Y_{-1})$. Using the induction hypothesis, $\delta(X_{-1}, Y_{-1}) = (\sum_{i=2\dots r+1} |x_i - y_i|) + |b_x - b_y|$. Therefore, it follows that $\delta(X, Y) = (\sum_{i=1\dots r+1} |x_i - y_i|) + |b_x - b_y|$ and the result holds. Now consider the case where $x_1 \neq y_1$. Without loss of generality, we will assume that $x_1 > y_1 \geq 1$. We can flip $x_1 - y_1$ bits in the sequence $x_1 + 1, x_1, x_1 - 1, \dots, y_1 + 1$ in the haplotype X until the first frontiers of both haplotypes match. All these moves do not change the number of frontiers of X and therefore are valid moves. After this sequence of moves, we can again consider the two haplotypes $X_{-1} = (x_2, \dots, x_{r+1})$ and $Y_{-1} = (y_2, \dots, y_{r+1})$ each with r frontiers. Using the induction hypothesis,

$$\delta(X, Y) = |x_1 - y_1| + \delta(X_{-1}, Y_{-1}) = \left(\sum_{i=1\dots r+1} |x_i - y_i| \right) + |b_x - b_y|$$

which proves the required result. ♣

Lemma 33:

$$\max_{X, Y \in D^k} \delta(X, Y) \leq (n - 2)k + 1$$

Proof: The maximum value of $|x_i - y_i|$ cannot exceed $n - 2$ for a haplotype of length n . Therefore, the sum $(\sum_{i=1\dots k} |x_i - y_i|)$ cannot exceed $(n - 2)k$. The result follows.



Now, we will construct a coupling $(X_t, Y_t) \rightarrow (X_{t+1}, Y_{t+1})$ for any pair of haplotypes (X_t, Y_t) where $X_t, Y_t \in D^k$ such that the expected distance $E(\delta(X_{t+1}, Y_{t+1}))$ is no less than $\delta(X_t, Y_t)$ and in addition, $\text{var}(\delta(X_{t+1}, Y_{t+1}))$ is sufficiently large. This allows us to use the bound on the mixing time for the case $\beta = 1$.

Theorem 34 For any pair of haplotypes $X_t, Y_t \in D^k$, there exists a coupling $(X_t, Y_t) \rightarrow (X_{t+1}, Y_{t+1})$ such that $E(\delta(X_{t+1}, Y_{t+1})) \geq \delta(X_t, Y_t)$ and $\text{Pr}[\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)] \geq \frac{1}{2(n+1)}$.

Proof: Let k_1 be the number of blocks of X_t of size 2 or more and k_2 be the number of blocks of Y_t of length 2 or greater. We assume that $k_1 > k_2$ without loss of generality. As we saw before, there are $2k_1 + 1$ valid moves for any haplotype $X_t \in D^k$ where $0 \leq k_1 \leq k$. Each such move has transition probability $\frac{1}{2(n+1)}$. With probability $1 - \frac{2 \cdot k_1 + 1}{2n+2}$, the haplotype X_t does not change. One can think of this probability as $2n - 2 \cdot k_1 + 1$ moves, each with probability $\frac{1}{2n+2}$ that take X_t to X_t . This notion of splitting the probability will be useful to describe the coupling. For every pair (X_t, Y_t) , we will add $2k_1 + 1$ transitions to the coupling that change X_t . Some of these will be coupled with moves that also change Y_t , others will be coupled with moves that do not change Y_t . Similarly, the coupling will have $2k_2 + 1$ moves for which $Y_{t+1} \neq Y_t$. All transitions in the coupling will have probability $\frac{1}{2n+2}$.

We first define the coupling transition for the bits b_x and b_y as follows: If $b_x = b_y$, we add the two transitions $(X_t, Y_t) \rightarrow (X_t, Y_t)$ and $(X_t, Y_t) \rightarrow (X_t(1 - b_x), Y_t(1 - b_y))$. If $b_x \neq b_y$, we add the two transitions $(X_t, Y_t) \rightarrow (X_t(1 - b_x), Y_t)$ and $(X_t, Y_t) \rightarrow (X_t, Y_t(1 - b_y))$. Clearly, $\delta(X_t(1 - b_x), Y_t) + \delta(X_t, Y_t(1 - b_y)) = 2 \times \delta(X_t, Y_t)$, therefore the expected distance is unchanged. Also, $\text{Pr}[\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)] \geq \frac{1}{2(n+1)}$.

We first consider the first block $(0, x_1 - 1)$ in the haplotype X_t and the corresponding block of Y_t . If the block of X_t is of length 1 and the block of Y_t has length at least 2, we add the following transition to the coupling: $(X_t, Y_t) \rightarrow (X_t, Y_t(1^-))$.

If the block of Y_t is of length 1 and the length of X_t is greater than 1, we add the move $(X_t, Y_t) \rightarrow (X_t(1^-), Y_t)$ to the coupling. In both cases, the following is true: $\delta(X_{t+1}, Y_{t+1}) < \delta(X_t, Y_t)$ and $Pr[\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)] = \frac{1}{2(n+1)}$. If both blocks are of length at least 2, we add the transition $(X_t, Y_t) \rightarrow (X_t(1^-), Y_t(1^-))$ to the coupling. We can similarly define transitions for the last block of X_t and Y_t . Transitions for all other pairs of blocks are defined according to the following rules: Consider a block $(x_i, x_{i+1} - 1)$ of X_t and let $(y_i, y_{i+1} - 1)$ be the corresponding block of Y_t .

Case I: $y_{i+1} = y_i + 1$.

Independent of the position of the block $(y_i, y_{i+1} - 1)$, flipping x_i and $x_{i+1} - 1$ will always have the opposite effect on δ . We will add two transitions to the coupling: i) $(X_t, Y_t) \rightarrow (X_t(i^+), Y_t)$ and ii) $(X_t, Y_t) \rightarrow (X_t((i+1)^-), Y_t)$. The expectation $E(\delta(X_{t+1}, Y_{t+1})) = \delta(X_t, Y_t)$ for the two transitions combined together and $Pr[\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)] = \frac{2}{2(n+1)}$.

Case II: $x_{i+1} = x_i + 1$.

Using symmetry, we can define the coupling transitions based on Case I above.

Case III: $y_i < x_i$ and $y_{i+1} < x_{i+1}$ and $y_{i+1} > y_i + 1$.

As before, flipping the position x_i will increase the distance between X_t and Y_t and flipping $x_{i+1} - 1$ will decrease δ . Flipping y_i decreases δ while flipping $y_{i+1} - 1$ increases δ . We will add the following two transitions to the coupling: i) $(X_t, Y_t) \rightarrow (X_t(i^+), Y_t((i+1)^-))$ and ii) $(X_t, Y_t) \rightarrow (X_t((i+1)^-), Y_t((i)^+))$. For the first transition, $\delta(X_{t+1}, Y_{t+1}) = \delta(X_t, Y_t) + 2$ and for the other, $\delta(X_{t+1}, Y_{t+1}) = \delta(X_t, Y_t) - 2$. Therefore, the expectation $E(\delta(X_{t+1}, Y_{t+1})) = \delta(X_t, Y_t)$ for the two transitions combined together and $Pr[\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)] = \frac{2}{2(n+1)}$.

Case IV: $y_i > x_i$ and $y_{i+1} > x_{i+1}$ and $y_{i+1} > y_i + 1$.

Using symmetry we can define coupling transitions for this case similarly to case II.

Case V: $y_i < x_i$ and $y_{i+1} > x_{i+1}$ and $y_{i+1} > y_i + 1$.

We will add the following two transitions to the coupling:

i) $(X_t, Y_t) \rightarrow (X_t(i^+), Y_t(i^+))$ and ii) $(X_t, Y_t) \rightarrow (X_t((i+1)^-), Y_t((i+1)^-))$.

For both transitions, $\delta(X_{t+1}, Y_{t+1}) = \delta(X_t, Y_t)$.

Case VI: $y_i > x_i$ and $y_{i+1} < x_{i+1}$ and $y_{i+1} > y_i + 1$.

Using symmetry we can define coupling transitions for this case similarly to case V.

Case VII: $y_i = x_i$ and $y_{i+1} \leq x_{i+1}$ and $y_{i+1} > y_i + 1$:

We again add two transitions: $(X_t, Y_t) \rightarrow (X_t(i^+), Y_t(i^+))$ and $(X_t, Y_t) \rightarrow (X_t((i+1)^-), Y_t((i+1)^-))$. For both transitions, $\delta(X_{t+1}, Y_{t+1}) = \delta(X_t, Y_t)$.

Case VIII: $y_i = x_i$ and $y_{i+1} > x_{i+1}$ and $y_{i+1} > y_i + 1$

Using symmetry we can define coupling transitions for this case similarly to case VII

We have defined 2 coupling transitions for every block of X_t (and Y_t) of size 2 or greater. For blocks in X_t of size 1 whose corresponding block in Y_t is also of size 1, we can define two identical coupling transitions: $(X_t, Y_t) \rightarrow (X_t, Y_t)$. In total, we will have $2k$ coupling transitions for (X_t, Y_t) each with probability $\frac{1}{2(n+1)}$. We add $2n - 2k$ coupling transitions of the form $(X_t, Y_t) \rightarrow (X_t, Y_t)$ each with probability $\frac{1}{2(n+1)}$. The coupling transitions for (X_t, Y_t) sum to 1. It is also easy to verify that there are $2k_1 + 1$ transitions for X_t alone such that $X_{t+1} \neq X_t$. Hence, marginally, both X_t and Y_t represent exact replicas of the Markov chain \mathcal{M}_k .

Now, we will analyze the expectation and variance of the distance function in one move of the coupling. In all cases, we have defined the coupling moves in pairs to ensure that $E(\delta(X_{t+1}, Y_{t+1})) \geq \delta(X_t, Y_t)$. In all the following cases, $Pr[\delta(X_{t+1}, Y_{t+1}) \neq$

$\delta(X_t, Y_t)] \geq \frac{1}{2^{(n+1)}}$ since there is at least one transition with probability $\frac{1}{2^{(n+1)}}$ that changes the distance between (X_{t+1}, Y_{t+1})

1. There exists a pair of blocks $(x_i, x_{i+1} - 1)$ of X_t and $(y_i, y_{i+1} - 1)$ of Y_t such that one of them has size 1 and the other one has size at least 2
2. $b_x \neq b_y$
3. There exists a pair of blocks $(x_i, x_{i+1} - 1)$ of X_t and $(y_i, y_{i+1} - 1)$ of Y_t such that $(y_i < x_i$ and $y_{i+1} < x_{i+1})$ or $(y_i > x_i$ and $y_{i+1} > x_{i+1})$

Therefore, the pairs (X_t, Y_t) left to analyze are those which do not satisfy any of the above three conditions.

Lemma 35: For all pairs of haplotypes (X_t, Y_t) such that $X_t, Y_t \in D^k$, $X_t \neq Y_t$ and the pair do not satisfy any of the three conditions listed above, there exist two non-trivial (length ≥ 2) blocks $(x_i, x_{i+1} - 1)$ and $(x_j, x_{j+1} - 1)$ in X_t that satisfy the following conditions:

- $x_{i+1} < x_j$
- $y_{i+1} > y_i + 1$ and $y_{j+1} - y_j > 1$
- $x_i \neq y_i$ or $x_{i+1} \neq y_{i+1}$
- $y_j \neq y_j$ or $x_{j+1} \neq y_{j+1}$
- $\delta(X_t((i+1)^-), Y_t) + \delta(X_t(j^-), Y_t) = 2 \cdot \delta(X_t, Y_t)$

Proof: A trivial block is a block of size 1 and a non-trivial one with size at least 2. Clearly, $k > 0$ since $b_x \neq b_y$ and there are only two haplotypes in D^0 . It is also easy to see that there is at least one non-trivial block in X_t otherwise $X_t = Y_t$ and $k = n - 2$. Let $(x_l, x_{l+1} - 1)$ be the first non-trivial block in X_t starting from the left. The corresponding block $(y_l, y_{l+1} - 1)$ of Y_t is also non-trivial otherwise the pair would satisfy the first condition listed above. Clearly, $x_l = y_l$ as all blocks before this one are

of size 1. Now, if $x_{l+1} = y_{l+1}$, we ignore this block and consider the first non-trivial block $(x_i, x_{i+1} - 1)$ starting from the left such that $x_{i+1} \neq y_{i+1}$. Such a block exists since otherwise the pair violates one of the three conditions above or $X_t = Y_t$. Now, since no trivial block in one haplotype can be matched to a non-trivial block in the other, there exists another pair of non-trivial blocks: $(x_j, x_{j+1} - 1)$ and $(y_j, y_{j+1} - 1)$ such that the pair is not perfectly aligned, i.e. $y_j \neq y_j$ or $x_{j+1} \neq y_{j+1}$. Furthermore, if $x_{i+1} < y_{i+1}$, then $x_j < y_j$ and vice versa. This implies that flipping $x_{i+1} - 1$ and x_j has opposite effect on the distance δ . Hence, $\delta(X_t((i+1)^-), Y_t) + \delta(X_t(j^-), Y_t) = 2 \cdot \delta(X_t, Y_t)$.

♣

We have established that for all pairs (X_t, Y_t) for which the coupling does not guarantee the condition: $Pr[\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)] \geq \frac{1}{2(n+1)}$, there are non-trivial blocks $(x_i, x_{i+1} - 1)$ and $(x_j, x_{j+1} - 1)$ in X_t whose corresponding blocks in Y_t are also non-trivial and satisfy certain conditions listed above. For such pairs, according to the rules of the coupling defined above, we added the transitions $(X_t, Y_t) \rightarrow (X_t((i+1)^-), Y_t((i+1)^-)$ and $(X_t, Y_t) \rightarrow (X_t(j^+), Y_t(j^+))$ to the coupling for these two blocks. Each transition independently does not change the distance δ . Let us assume that $\delta(X_t((i+1)^-), Y_t) = \delta(X_t, Y_t) + 1$ and $\delta(X_t, Y_t((i+1)^-)) = \delta(X_t, Y_t) - 1$. It follows that $\delta(X_t((j)^+), Y_t) = \delta(X_t, Y_t) - 1$ and $\delta(X_t, Y_t((j)^+)) = \delta(X_t, Y_t) + 1$. We will remove the two transitions $(X_t, Y_t) \rightarrow (X_t((i+1)^-), Y_t((i+1)^-)$ and $(X_t, Y_t) \rightarrow (X_t(j^+), Y_t(j^+))$ and replace them by the two transitions: i) $(X_t, Y_t) \rightarrow (X_t((i+1)^-), Y_t(j^+))$ and, ii) $(X_t, Y_t) \rightarrow (X_t(j^+), Y_t((i+1)^-))$. For these two new transitions, $(\delta(X_{t+1}, Y_{t+1})) = \delta(X_t, Y_t)$ and $Pr[\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)] \geq \frac{1}{2(n+1)}$.

♣

In our modified coupling, for all pairs of haplotypes (X_t, Y_t) , $(\delta(X_{t+1}, Y_{t+1})) \geq \delta(X_t, Y_t)$ and $Pr[\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)] \geq \frac{1}{2(n+1)}$. Hence, we can apply the bound for the case $\beta = 1$ on the mixing time:

$$\tau(\epsilon) \leq \lceil \frac{eD^2}{\alpha} \rceil \lceil \log(\epsilon^{-1}) \rceil \leq c(n+1)^3 k^2 \lceil \log(\epsilon^{-1}) \rceil$$

where c is a small constant. In the worst case, the bound is $O(n^5 \log(\epsilon^{-1}))$.



7.4.2 Mixing time of the projection Markov chain

The Markov chain that moves between the subsets $\Omega_1, \Omega_2, \dots, \Omega_m$ is called the projection Markov chain \mathcal{M}_H and is defined on the set $[m] = \{1, 2, \dots, m\}$ where each point i corresponds to the set Ω_i . The transition matrix $P_H(i, j)$ is defined as

$$P_H(i, j) = \frac{1}{\pi(\Omega_i)} \sum_{x \in \Omega_i, y \in \Omega_j} \pi(x) P(x, y)$$

For our Markov chain, $m = n - 1$ and $\Omega_1 = D^0, \Omega_2 = D^1, \dots, \Omega_{n-1} = D^{n-2}$. Consider two adjacent subsets D^{k-1} and D^k . Every haplotype in D^k for which $x_1 = 1$ or $x_k = n - 1$ has a neighbor in the set D^{k-1} . The number of such haplotypes is

$$2 \binom{n-3}{k-1} + 2 \binom{n-3}{k-1} - 2 \binom{n-4}{k-2} \geq 2 \binom{n-3}{k-1}$$

Therefore

$$Q((D^k, D^{k-1})) \geq \pi(H^1) \rho^k 2 \cdot \binom{n-3}{k-1} \frac{1}{2(n+1)} \geq \binom{n-3}{k-1} \frac{\pi(H^1) \rho^k}{n+1}$$

Similarly,

$$Q((D^{k-1}, D^k)) \geq \pi(H^1) \rho^{k-1} 2 \cdot \binom{n-3}{k-1} \frac{\rho}{2(n+1)} \geq \binom{n-3}{k-1} \frac{\pi(H^1) \rho^k}{n+1}$$

We will use the canonical path argument to bound the mixing time of this Markov chain. The canonical path from D^i to D^j where $j > i$ will use the neighboring edges $(D^k, D^{k+1}), i \leq k < j$ and has length $j - i$. Similarly, the canonical path from D_j to D^i uses the edges $(D^k, D^{k-1}), j \leq k < i$ and also has length $j - i$. Note that there are also edges between D^k and D^{k-2} in the Markov chain, but we do not use these edges and hence the congestion on these edges is zero. Consider an edge $e = (D^k, D^{k-1})$. We

will upper bound the congestion on this edge using the above inequalities.

$$\begin{aligned}
\bar{\rho}(e) &= \frac{1}{Q(e)} \sum_{\gamma_{xy} \ni t} \pi(x)\pi(y)|\gamma_{xy}| \\
&\leq \frac{(n+1)}{\pi(H^1)\rho^k \binom{n-3}{k-1}} \sum_{x=0 \dots k-1, y=k \dots n-2} 4\pi(H^1)^2 \rho^x \rho^y \binom{n-2}{x} \binom{n-2}{y} (y-x) \\
&\leq 4(n+1) \cdot \pi(H^1) \left[\sum_{x=0 \dots k-1} \rho^x \binom{n-2}{x} \right] \left[\sum_{y=k \dots n-2} \frac{\rho^y \binom{n-2}{y}}{\rho^k \binom{n-3}{k-1}} \right] \cdot n \\
&\leq 2(n+1) \cdot \left[\sum_{x=0 \dots k-1} 2\pi(H^1)\rho^x \binom{n-2}{x} \right] \left[\sum_{y=k \dots n-2} \frac{\rho^y \binom{n-2}{y}}{\rho^k \binom{n-3}{k-1}} \right] \cdot n \\
&\leq 2(n+1) \cdot [\pi(D^0) + \pi(D^1) + \dots + \pi(D^{k-1})] \left[\sum_{y=k \dots n-2} \frac{\rho^y \binom{n-2}{y}}{\rho^k \binom{n-3}{k-1}} \right] \cdot n \\
&\leq 2(n+1) \left[\sum_{y=k \dots n-2} \frac{\rho^y \binom{n-2}{y}}{\rho^k \binom{n-3}{k-1}} \right] \cdot n \\
&\leq 2(n+1) \left[\frac{n-2}{k} + \frac{(n-2)(1 - \frac{k-2}{n})}{k(k+1)} \rho \cdot n + \dots \right] \cdot n \\
&\leq 2(n+1)(n-2) \left[\frac{1}{k} + \frac{1}{k(k+1)} + \frac{1}{k(k+1)(k+2)} + \dots \right] \cdot n \\
&\leq 4n^3
\end{aligned}$$

The congestion on the edge $e' = (D^{k-1}, D^k)$ can be also be bounded identically since $Q(e)$ is same for both edges. Therefore, it follows that

$$\begin{aligned}
\tau_H(\epsilon) &\leq 4n^3 (\ln(\epsilon^{-1}) + \ln(\pi_{min}^{-1})) \leq 4n^3 \left(\ln(\epsilon^{-1}) + \ln \left(\frac{1}{\rho^{n-2}} \right) \right) \\
&\Rightarrow \tau_H(\epsilon) \leq 4n^3 (\ln(\epsilon^{-1}) + n \ln(\rho^{-1}))
\end{aligned}$$

We are now ready to prove Theorem 30.

Proof:

We have the following inequality:

$$4n^3 (\ln(\epsilon^{-1}) + n \ln(\rho^{-1})) \geq \tau_H(\epsilon) \geq \frac{1}{2\text{Gap}(P_H)} \log \left(\frac{1}{2\epsilon} \right)$$

The above inequality is true for all $0 < \epsilon < 1$. Choosing $\epsilon = \frac{1}{2\rho}$, we get

$$\text{Gap}(P_H) \geq \frac{c_1}{n^4}$$

for some constant c_1 . We have also shown that for each of the Markov chains \mathcal{M}_k ,

$$\tau_H(\epsilon) \leq c(n+1)^3 k^2 \ln(\epsilon^{-1})$$

It follows that

$$\text{Gap}(\mathcal{M}_k) \geq \frac{c_2}{n^5}$$

for some constant c_2 and all k . Using Theorem 11, we have

$$\text{Gap}(P) \geq \frac{1}{2} \text{Gap}(P_H) \min_{i=1 \dots m} \text{Gap}(P_{\Omega_i}) \geq \frac{1}{2} \frac{c_1}{n^4} \frac{c_2}{n^5} \geq \frac{c_3}{n^9}$$

for some constant c_3 . Finally, the mixing time of the complete Markov chain \mathcal{M} can be bounded as follows

$$\tau_{\mathcal{M}}(\epsilon) \leq \frac{1}{\text{Gap}(P)} \ln \left(\frac{1}{\min_x \pi(x)} \cdot \epsilon \right) \leq \frac{n^9}{c^3} (n \ln(\rho^{-1}) + \ln(\epsilon^{-1}))$$



References

- Aldous, D., 1986: Random walks on finite groups and rapidly mixing Markov chains, *Seminaire de Probabilites XVII 1981/82* (A. Dold and B. Eckmann, eds). *Springer Lecture Notes in Mathematics*, **986**, 243–297.
- Andolfatto, P., Depaulis, F., and Navarro, A., 2001: Inversion polymorphisms and nucleotide variability in *Drosophila*. *Genet Res*, **77**(1), 1–8.
- Bafna, V., and Bansal, V., 2005: Improved recombination lower bounds for haplotype data. In *RECOMB*, editors S. Miyano, J. P. Mesirov, S. Kasif, S. Istrail, P. A. Pevzner, and M. S. Waterman, volume 3500 of *Lecture Notes in Computer Science*, 569–584. Springer. ISBN 3-540-25866-3.
- Bafna, V., Gusfield, D., Lancia, G., and Yooseph, S., 2003: Haplotyping as a perfect phylogeny. A direct approach. *Journal of Computational Biology*. To appear.
- Bafna, V., Istrail, S., Lancia, G., and Rizzi, R., 2005: Polynomial and APX-hard cases of Individual Haplotyping Problems. *Theoretical Computer Science*, **335**(1), 109–125.
- Bagnall, R. D., Waseem, N., Green, P. M., and Giannelli, F., 2002: Recurrent inversion breaking intron 1 of the factor viii gene is a frequent cause of severe hemophilia a. *Blood*, **99**(1), 168–174.
- Bentley, D., 2006: Whole-genome re-sequencing. *Curr. Opin. Genet. Dev*, **16**, 545–552.
- Broman, K., Matsumoto, N., Giglio, S., Martin, C., Roseberry, J., Zuffardi, O., Ledbetter, D., and Weber, J., 2003: *Science and Statistics: A Festschrift for Terry Speed*, volume 40, chapter Common long human inversion polymorphism on chromosome 8p, 237–245. IMS Lecture Notes-Monograph Series.
- Broman, K. W., Murray, J. C., Sheffield, V. C., White, R. L., and Weber, J. L., 1998: Comprehensive human genetic maps: individual and sex-specific variation in recombination. *Am J Hum Genet*, **63**(3), 861–869.
- Cilibrasi, R., van Iersel, L., Kelk, S., and Tromp, J., 2005: On the complexity of several haplotyping problems. In *WABI*, editors R. Casadio, and G. Myers, volume 3692 of *Lecture Notes in Computer Science*, 128–139. Springer. ISBN 3-540-29008-7.

- Clark, A., Weiss, K., Nickerson, D., Taylor, S., Buchanan, A., Stengard, J., Salomaa, V., Vartiainen, E., Perola, M., Boerwinkle, E., and Sing, C., 1998: Haplotype structure and population genetic inferences from nucleotide-sequence variation in human lipoprotein lipase. *American Journal of Human Genetics*, **63**, 595–612.
- Clark, A. G., 1990: Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular Biology and Evolution*, **7**(2), 111–122.
- Conrad, D., Andrews, T., Carter, N., Hurles, M., and Pritchard, J., 2006: A high-resolution survey of deletion polymorphism in the human genome. *Nat Genet*, **38**(1), 75–81. doi:10.1038/ng1697.
- Crawford, D., Bhangale, T., Li, N., Hellenthal, G., Rieder, M., Nickerson, D., and Stephens, M., 2004: Evidence for substantial fine-scale variation in recombination rates across the human genome. *Nature Genetics*, **36**, 700–706.
- Daly, M. J., Rioux, J. D., Schaffner, S. F., Hudson, T. J., and Lander, E. S., 2001: High-resolution haplotype structure in the human genome. *Nature Genetics*, **29**, 229–232.
- Deutz-Terlouw, P. P., Losekoot, M., Olmer, R., Pieneman, W. C., de Vries-v d Weerd, S., Briët, E., and Bakker, E., 1995: Inversions in the factor viii gene: improvement of carrier detection and prenatal diagnosis in dutch haemophilia a families. *J Med Genet*, **32**(4), 296–300.
- D.Gusfield, Eddhu, S., and C.Langley, 2003: Efficient reconstruction of phylogenetic networks with constrained recombination. In *IEEE Computer Society Bioinformatics Conference*, 363–374.
- D.Gusfield, Eddhu, S., and C.Langley, 2004: The fine structure of galls in phylogenetic networks. In *To Appear in INFORMS J. of Computing: Special Issue on Computational Biology*.
- Eskin, E., Halperin, E., and Karp, R., 2003: Efficient reconstruction of haplotype structure via perfect phylogeny. *Journal of Bioinformatics and Computational Biology*, **1**, 1–20.
- Ewing, B., and Green, P., 1998: Base-calling of automated sequencer traces using phred. II. Error probabilities. *Genome Res*, **8**(3), 186–194.
- Excoffier, L., and Slatkin, M., 1995: Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Molecular Biology and Evolution*, **12**(5), 921–927.
- Fearnhead, P., and Donnelly, P., 2001: Estimating recombination rates from population genetic data. *Genetics*, **159**, 1299–1318.

- Fearnhead, P., Harding, R., Schneider, J., Myers, S., and Donnelly, P., 2004: Application of coalescent methods to reveal fine-scale rate variation and recombination hotspots. *Genetics*, **167**, 2067–2081.
- Feuk, L., Macdonald, J., Tang, T., Carson, A., Li, M., Rao, G., Khaja, R., and Scherer, S., 2005: Discovery of human inversion polymorphisms by comparative analysis of human and chimpanzee DNA sequence assemblies. *PLoS Genet*, **1**(4). doi:10.1371/journal.pgen.0010056.
- Gabriel, S. B., Schaffner, S. F., Nguyen, H., Moore, J. M., Roy, J., Blumenstiel, B., Higgins, J., DeFelice, M., Lochner, A., Faggart, M., Liu-Cordero, S. N., Rotimi, C., Adeyemo, A., Cooper, R., Ward, R., Lander, E. S., Daly, M. J., and Altshuler, D., 2002: The structure of haplotype blocks in the human genome. *Science*, **296**(5576), 2225–2229.
- Gaedigk, R., Karges, W., Hui, M. F., Scherer, S. W., and Dosch, H. M., 1996: Genomic organization and transcript analysis of ICAp69, a target antigen in diabetic autoimmunity. *Genomics*, **38**(3), 382–391.
- Garey, M. R., and Johnson, D. S., 1979: *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman and Company.
- Giglio, S., Broman, K., Matsumoto, N., Calvari, V., Gimelli, G., Neumann, T., Ohashi, H., Voullaire, L., Larizza, D., Giorda, R., Weber, J., Ledbetter, D., and Zuffardi, O., 2001: Olfactory receptor-gene clusters, genomic-inversion polymorphisms, and common chromosome rearrangements. *Am J Hum Genet*, **68**(4), 874–883.
- Griffiths, R. C., and Marjoram, P., 1996: Ancestral inference from samples of DNA sequences with recombination. *Journal of Computational Biology*, **3**(4), 479–502.
- Gudmundsson, J., Sulem, P., Manolescu, A., Amundadottir, L., Gudbjartsson, D., Helgason, A., Rafnar, T., Bergthorsson, J., Agnarsson, B., Baker, A., Sigurdsson, A., Benediksdottir, K., Jakobsdottir, M., Xu, J., Blondal, T., Kostic, J., Sun, J., Ghosh, S., Stacey, S., Mouy, M., Saemundsdottir, J., Backman, V., Kristjansson, K., Tres, A., Partin, A., Albers-Akkers, M., Godino-Ivan Marcos, J., Walsh, P., Swinkels, D., Navarrete, S., Isaacs, S., Aben, K., Graif, T., Cashy, J., Ruiz-Echarri, M., Wiley, K., Suarez, B., Witjes, J., Frigge, M., Ober, C., Jonsson, E., Einarsson, G., Mayordomo, J., Kiemeny, L., Isaacs, W., Catalona, W., Barkardottir, R., Gulcher, J., Thorsteinsdottir, U., Kong, A., and Stefansson, K., 2007: Genome-wide association study identifies a second prostate cancer susceptibility variant at 8q24. *Nat. Genet.*, **39**, 631–637.
- Gusfield, D., 1991: Efficient algorithms for inferring evolutionary trees. *Networks*, **21**, 19–28.

- Gusfield, D., 2002: Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions (Extended abstract). In *Proceedings of the Sixth Annual International Conference on Computational Molecular Biology (RECOMB)*, 166–175.
- Halldorsson, B. V., Bafna, V., Edwards, N., Lippert, R., Yooseph, S., and Istrail, S., 2003: Combinatorial Problems Arising in SNP and Haplotype Analysis. In *DMTCS*, 26–47.
- Hedrick, P. W., 1987a: Gametic disequilibrium measures: proceed with caution. *Genetics*, **117**(2), 331–341.
- Hedrick, P. W., 1987b: Gametic disequilibrium measures: proceed with caution. *Genetics*, **117**(2), 331–341.
- Hein, J., 1990: Reconstructing Evolution of sequences subject to recombination using parsimony. *Math. Biosci.*, **98**, 185–200.
- Hein, J., 1993: A Heuristic Method to Reconstruct the History of Sequences Subject to Recombination. *J. Mol. Evol.*, **20**, 402–411.
- Hinds, D., Kloek, A., Jen, M., Chen, X., and Frazer, K., 2006: Common deletions and SNPs are in linkage disequilibrium in the human genome. *Nat Genet*, **38**(1), 82–85. doi:10.1038/ng1695.
- Hinds, D., Stuve, L., Nilsen, G., Halperin, E., Eskin, E., Ballinger, D., Frazer, K., and Cox, D., 2005: Whole-genome patterns of common DNA variation in three human populations. *Science*, **307**, 1072–1079.
- Hudson, R. R., 1990: Gene genealogies and the coalescent process. In *Oxford surveys in evolutionary biology*, editors D. Futuyma, and J. Antonovics, volume 7, 1–44. Oxford University Press.
- Hudson, R. R., 2001: Two-locus sampling distributions and their applications. *Genetics*, **159**, 1805–1817.
- Hudson, R. R., 2002: Generating samples under a wright-fisher neutral model of genetic variation. *Bioinformatics*, **18**(2), 337–338.
- Hudson, R. R., and Kaplan, N. L., 1985: Statistical properties of the number of recombination events in the history of a sample of DNA sequences. *Genetics*, **111**, 147–164.
- Iafrate, A. J., Feuk, L., Rivera, M. N., Listewnik, M. L., Donahoe, P. K., Qi, Y., Scherer, S. W., and Lee, C., 2004: Detection of large-scale variation in the human genome. *Nat Genet*, **36**(9), 949–951. doi:10.1038/ng1416.
- International Human Genome Sequencing Consortium, 2001: Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.

- Istrail, S., Sutton, G. G., Florea, L., Halpern, A. L., Mobarry, C. M., Lippert, R., Walenz, B., Shatkay, H., Dew, I., Miller, J. R., Flanigan, M. J., Edwards, N. J., Bolanos, R., Fasulo, D., Halldorsson, B. V., Hannenhalli, S., Turner, R., Yooseph, S., Lu, F., Nusskern, D. R., Shue, B. C., Zheng, X. H., Zhong, F., Delcher, A. L., Huson, D. H., Kravitz, S. A., Mouchard, L., Reinert, K., Remington, K. A., Clark, A. G., Waterman, M. S., Eichler, E. E., Adams, M. D., Hunkapiller, M. W., Myers, E. W., and Venter, J. C., 2004: Whole-genome shotgun assembly and comparison of human genome assemblies. *Proc Natl Acad Sci U S A*, **101**(7), 1916–1921. doi:10.1073/pnas.0307971100.
- Jeffreys, A., Ritchie, A., and Neumann, R., 2000: High resolution analysis of haplotype diversity and meiotic crossover in the human tap2 recombination hotspot. *Human Molecular Genetics*, **9**, 725–733.
- Jeffreys, A. J., Kauppi, L., and Neumann, R., 2001: Intensely punctate meiotic recombination in the class II region of the major histocompatibility complex. *Nature Genetics*, **29**(2), 217–222.
- Jeffreys, A. J., Neumann, R., Panayi, M., Myers, S., and Donnelly, P., 2005: Human recombination hot spots hidden in regions of strong marker association. *Nat Genet*, **37**(6), 601–606. doi:10.1038/ng1565.
- Jerrum, M., and Sinclair, A., 1997: The markov chain monte carlo method: an approach to approximate counting and integration. 482–520.
- Johnson, D., 1972: Approximation algorithms for combinatorial problems. *Journal of Comput. System Sci.*, **9**, 256–278.
- Kauppi, L., Sajantila, A., and Jeffreys, A. J., 2003: Recombination hotspots rather than population history dominate linkage disequilibrium in the mhc class ii region. *Hum Mol Genet*, **12**(1), 33–40.
- Kim, J., Waterman, M., and Li, L., 2007: Diploid genome reconstruction of *Ciona intestinalis* and comparative analysis with *Ciona savignyi*. *Genome Research*, **17**(7), 1101.
- Kimmel, G., and Shamir, R., 2004: The incomplete perfect phylogeny haplotype problem. *To appear in Journal of Bioinformatics and Computational Biology*.
- Kingman, J., 1982: The coalescent. *Stochastic Process Appl.*
- Konfortov, B., Bankier, A., and Dear, P., 2007: An efficient method for multi-locus molecular haplotyping. *Nucleic Acids Research*, **35**(1), e6.
- Kong, A., Gudbjartsson, D., Sainz, J., Jonsdottir, G., Gudjonsson, S., Richardsson, B., Sigurdardottir, S., Barnard, J., Hallbeck, B., Masson, G., Shlien, A., Palsson,

- S., Frigge, M., Thorgeirsson, T., Gulcher, J., and Stefansson, K., 2002a: A high-resolution recombination map of the human genome. *Nat Genet*, **31**(3), 241–247. doi:10.1038/ng917.
- Kong, A., Gudbjartsson, D. F., Sainz, J., Jonsdottir, G. M., Gudjonsson, S. A., Richardson, B., Sigurdardottir, S., Barnard, J., Hallbeck, B., Masson, G., Shlien, A., Palsson, S. T., Frigge, M. L., Thorgeirsson, T. E., Gulcher, J. R., and Stefansson, K., 2002b: A high-resolution recombination map of the human genome. *Nature Genetics*, **31**(3), 241–247.
- Kreitman, M., 1983: Nucleotide polymorphism at the alcohol dehydrogenase locus of *Drosophila melanogaster*. *Nature*, **304**, 412–417.
- Lakich, D., Kazazian, H. H., Antonarakis, S. E., and Gitschier, J., 1993: Inversions disrupting the factor VIII gene are a common cause of severe haemophilia A. *Nat Genet*, **5**(3), 236–241. doi:10.1038/ng1193-236.
- Lancia, G., Bafna, V., Istrail, S., Lippert, R., and Schwartz, R., 2001: SNPs problems, complexity and algorithms. In *Proceedings of the Ninth Annual European Symposium on Algorithms (ESA)*, 182–193.
- Levy, S., Sutton, G., Ng, P. C., Feuk, L., Halpern, A. L., Walenz, B. P., Axelrod, N., Huang, J., Kirkness, E. F., Denisov, G., Lin, Y., MacDonald, J. R., Pang, A. W., Shago, M., Stockwell, T. B., Tsiamouri, A., Bafna, V., Bansal, V., Kravitz, S. A., Busam, D. A., Beeson, K. Y., McIntosh, T. C., Remington, K. A., Abril, J. F., Gill, J., Borman, J., Rogers, Y. H., Frazier, M. E., Scherer, S. W., Strausberg, R. L., and Venter, J. C., 2007: The diploid genome sequence of an individual human. *PLoS Biol*, **5**(10). doi:10.1371/journal.pbio.0050254.
- Lewontin, R., 1964: The interaction of selection and linkage II. Optimum models. *Genetics*, **50**, 757–782.
- Li, L. M., Kim, J. H., and Waterman, M. S., 2004: Haplotype reconstruction from SNP alignment. *J Comput Biol*, **11**(2-3), 505–516. doi:10.1089/1066527041410454.
- Li, N., and Stephens, M., 2003: Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data. *Genetics*, **165**, 2213–2233.
- Lin, S., Cutler, D. J., Zwick, M. E., and Chakravarti, A., 2002: Haplotype inference in random population samples. *Am J Hum Genet*, **71**(5), 1129–1137.
- Lippert, R., Schwartz, R., Lancia, G., and Istrail, S., 2002: Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. *Briefings in Bioinformatics*, **3**(1), 23–31.

- Lucito, R., Healy, J., Alexander, J., Reiner, A., Esposito, D., Chi, M., Rodgers, L., Brady, A., Sebat, J., Troge, J., West, J. A., Rostan, S., Nguyen, K. C., Powers, S., Ye, K. Q., Olshen, A., Venkatraman, E., Norton, L., and Wigler, M., 2003: Representational oligonucleotide microarray analysis: a high-resolution method to detect genome copy number variation. *Genome Res*, **13**(10), 2291–2305. doi: 10.1101/gr.1349003.
- Lupski, J. R., 1998: Genomic disorders: structural features of the genome can lead to dna rearrangements and human disease traits. *Trends Genet*, **14**(10), 417–422.
- Madras, N., and Randall, D., 2002: Markov Chain Decomposition for Convergence Rate Analysis. *The Annals of Applied Probability*, **12**(2), 581–606.
- Marchini, J., Cutler, D., Patterson, N., Stephens, M., Eskin, E., Halperin, E., Lin, S., Qin, Z., Munro, H., Abecasis, G., and Donnelly, P., 2006: A comparison of phasing algorithms for trios and unrelated individuals. *Am. J. Hum. Genet.*, **78**, 437–450.
- Marchini, J., Howie, B., Myers, S., McVean, G., and Donnelly, P., 2007: A new multi-point method for genome-wide association studies by imputation of genotypes. *Nat. Genet.*, **39**, 906–913.
- Martin, R., and Randall, D., 2006: Disjoint decomposition of markov chains and sampling circuits in cayley graphs. *Comb. Probab. Comput.*, **15**(3), 411–448. ISSN 0963-5483. doi:http://dx.doi.org/10.1017/S0963548305007352.
- MARTIN, R., and RANDALL, D., 2006: Disjoint Decomposition of Markov Chains and Sampling Circuits in Cayley Graphs. *Combinatorics, Probability and Computing*, **15**(03), 411–448.
- McCarroll, S., Hadnott, T., Perry, G., Sabeti, P., Zody, M., Barrett, J., Dallaire, S., Gabriel, S., Lee, C., Daly, M., Altshuler, D., and Consortium, T. I. H., 2006: Common deletion polymorphisms in the human genome. *Nat Genet*, **38**(1), 86–92.
- McVean, G., Awadella, T., and Fearnhead, P., 2002: A coalescent method for detecting recombination from gene sequences. *Genetics*, **160**, 1231–1241.
- McVean, G., Myers, S., Hunt, S., Deloukas, P., Bentley, D., and Donnelly, P., 2004: The fine-scale structure of recombination rate variation in the human genome. *Science*, **304**, 581–584.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E., 1953: Equations of state calculations by fast computing machine. *J. Chem. Phys.*, **21**, 1087–1091.
- Myers, S., Bottolo, L., Freeman, C., McVean, G., and Donnelly, P., 2005: A fine-scale map of recombination rates and hotspots across the human genome. *Science*, **310**(5746), 321–324. doi:10.1126/science.1117196.

- Myers, S., and Griffiths, R., 2003: Bounds on the Minimum Number of Recombination Events in a Sample History. *Genetics*, **163**, 375–394.
- Navarro, A., Barbadilla, A., and Ruiz, A., 2000a: Effect of inversion polymorphism on the neutral nucleotide variability of linked chromosomal regions in drosophila. *Genetics*, **155**(2), 685–698.
- Navarro, A., Barbadilla, A., and Ruiz, A., 2000b: Effect of inversion polymorphism on the neutral nucleotide variability of linked chromosomal regions in Drosophila. *Genetics*, **155**(2), 685–698.
- Navarro, A., Betrán, E., Barbadilla, A., and Ruiz, A., 1997: Recombination and gene flux caused by gene conversion and crossing over in inversion heterokaryotypes. *Genetics*, **146**(2), 695–709.
- Navarro, A., and Gazave, E., 2005: Inversions with classical style and trendy lines. *Nat Genet*, **37**(2), 115–116. doi:10.1038/ng0205-115.
- Newman, T. L., Tuzun, E., Morrison, V. A., Hayden, K. E., Ventura, M., McGrath, S. D., Rocchi, M., and Eichler, E. E., 2005: A genome-wide survey of structural variation between human and chimpanzee. *Genome Res*, **15**(10), 1344–1356. doi:10.1101/gr.4338005.
- Nickerson, D., Taylor, S., Weiss, K., Clark, A., Hutchinson, R., Stengard, J., Salomaa, V., Vartiainen, E., Boerwinkle, E., and Sing, C., 1998: Dna sequence diversity in a 9.7-kb region of the human lipoprotein lipase gene. *Nature Genetics*, **19**, 233–240.
- Niu, T., Qin, Z. S., Xu, X., and Liu, J. S., 2002: Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *American Journal of Human Genetics*, **70**, 157–169.
- Osborne, L., Li, M., Pober, B., Chitayat, D., Bodurtha, J., Mandel, A., Costa, T., Grebe, T., Cox, S., Tsui, L., and Scherer, S., 2001: A 1.5 million-base pair inversion polymorphism in families with Williams-Beuren syndrome. *Nat Genet*, **29**(3), 321–325. doi:10.1038/ng753.
- Panconesi, A., and Sozio, M., 2004: Fast hare: A fast heuristic for single individual snp haplotype reconstruction. In *WABI*, editors I. Jonassen, and J. Kim, volume 3240 of *Lecture Notes in Computer Science*, 266–277. Springer. ISBN 3-540-23018-1.
- Pe'er, I., de Bakker, P., Maller, J., Yelensky, R., Altshuler, D., and Daly, M., 2006: Evaluating and improving power in whole-genome association studies using fixed marker sets. *Nat. Genet.*, **38**, 663–667.
- Pritchard, J. K., and Przeworski, M., 2001: Linkage disequilibrium in humans: models and data. *Am J Hum Genet*, **69**(1), 1–14.

- Ptak, S. E., Hinds, D. A., Koehler, K., Nickel, B., Patil, N., Ballinger, D. G., Przeworski, M., Frazer, K. A., and Pääbo, S., 2005: Fine-scale recombination patterns differ between chimpanzees and humans. *Nat Genet*, **37**(4), 429–434. doi:10.1038/ng1529.
- Ptak, S. E., Roeder, A. D., Stephens, M., Gilad, Y., Pääbo, S., and Przeworski, M., 2004: Absence of the tap2 human recombination hotspot in chimpanzees. *PLoS Biol*, **2**(6). doi:10.1371/journal.pbio.0020155.
- Randall, D., 2006: Rapidly mixing markov chains with applications in computer science and physics. *Computing in Science and Engineering*, **8**(2), 30–41.
- Rizzi, R., Bafna, V., Istrail, S., and Lancia, G., 2002: Practical algorithms and fixed-parameter tractability for the single individual SNP haplotyping problem. In *Proceedings of the Second International Workshop on Algorithms in Bioinformatics (WABI)*, 29–43.
- Rosenberg, N. A., and Nordborg, M., 2002: Genealogical trees, coalescent theory and the analysis of genetic polymorphisms. *Nat Rev Genet*, **3**(5), 380–390. doi:10.1038/nrg795.
- Sachidanandam, R., Weissman, D., Schmidt, S. C., Kakol, J. M., Stein, L. D., Marth, G., Sherry, S., Mullikin, J. C., Mortimore, B. J., Willey, D. L., Hunt, S. E., Cole, C. G., Coggill, P. C., Rice, C. M., Ning, Z., Rogers, J., Bentley, D. R., Kwok, P. Y., Mardis, E. R., Yeh, R. T., Schultz, B., Cook, L., Davenport, R., Dante, M., Fulton, L., Hillier, L., Waterston, R. H., McPherson, J. D., Gilman, B., Schaffner, S., Van Etten, W. J., Reich, D., Higgins, J., Daly, M. J., Blumenstiel, B., Baldwin, J., Stange-Thomann, N., Zody, M. C., Linton, L., Lander, E. S., Altshuler, D., and , T. I. S. M. W. G., 2001: A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. *Nature*, **409**(6822), 928–933. doi:10.1038/35057149.
- Sahni, S., and Gonzalez, T., 1974: P-complete problems and approximate solutions. In *FOCS*, 28–32.
- Sampath, J., Long, P. R., Shepard, R. L., Xia, X., Devanarayan, V., Sandusky, G. E., Perry, W. L., Dantzig, A. H., Williamson, M., Rolfe, M., and Moore, R. E., 2003: Human spf45, a splicing factor, has limited expression in normal tissues, is overexpressed in many tumors, and can confer a multidrug-resistant phenotype to cells. *Am J Pathol*, **163**(5), 1781–1790.
- Schaffner, S. F., Foo, C., Gabriel, S., Reich, D., Daly, M. J., and Altshuler, D., 2005: Calibrating a coalescent simulation of human genome sequence variation. *Genome Res*, **15**(11), 1576–1583. doi:10.1101/gr.3709305.
- SCHULTZ, J., and REDFIELD, H., 1951: Interchromosomal effects on crossing over in drosophila. *Cold Spring Harb Symp Quant Biol*, **16**, 175–197.

- Schuster, S., 2008: Next-generation sequencing transforms today's biology. *Nat. Methods*, **5**, 16–18.
- Sebat, J., Lakshmi, B., Troge, J., Alexander, J., Young, J., Lundin, P., Månér, S., Massa, H., Walker, M., Chi, M., Navin, N., Lucito, R., Healy, J., Hicks, J., Ye, K., Reiner, A., Gilliam, T. C., Trask, B., Patterson, N., Zetterberg, A., and Wigler, M., 2004: Large-scale copy number polymorphism in the human genome. *Science*, **305**(5683), 525–528. doi:10.1126/science.1098918.
- Shaffer, C., 2007: Next-generation sequencing outpaces expectations. *Nat. Biotechnol.*, **25**, 149.
- Sharp, A. J., Locke, D. P., McGrath, S. D., Cheng, Z., Bailey, J. A., Vallente, R. U., Pertz, L. M., Clark, R. A., Schwartz, S., Segraves, R., Oseroff, V. V., Albertson, D. G., Pinkel, D., and Eichler, E. E., 2005: Segmental duplications and copy-number variation in the human genome. *Am J Hum Genet*, **77**(1), 78–88. doi:10.1086/431652.
- Shaw, C., and Lupski, J., 2004: Implications of human genome architecture for rearrangement-based disorders: the genomic basis of disease. *Hum Mol Genet*, **13 Spec No 1**, 57–64. doi:10.1093/hmg/ddh073.
- Sherry, S. T., Ward, M. H., Kholodov, M., Baker, J., Phan, L., Smigielski, E. M., and Sirotkin, K., 2001: dbSNP: the ncbi database of genetic variation. *Nucleic Acids Res*, **29**(1), 308–311.
- Sinclair, A., 1992: Improved Bounds for Mixing Rates of Markov Chains and Multi-commodity Flow. *Combinatorics, Probability & Computing*, **1**, 351–370.
- Sinclair, A., and Jerrum, M., 1989: Approximate counting, uniform generation and rapidly mixing markov chains. *Inf. Comput.*, **82**(1), 93–133. ISSN 0890-5401. doi: [http://dx.doi.org/10.1016/0890-5401\(89\)90067-9](http://dx.doi.org/10.1016/0890-5401(89)90067-9).
- Song, Y., and Hein, J., 2003: Parsimonious Reconstruction of Sequence Evolution and Haplotype Blocks: Finding the Minimum Number of Recombination Events. In *In Algorithms in Bioinformatics, WABI 2003*, 287–302.
- Song, Y., and Hein, J., 2004: On the minimum number of recombination events in the evolutionary history of dna sequences. *Journal of Mathematical Biology*, **48**, 160–186.
- Stefansson, H., Helgason, A., Thorleifsson, G., Steinthorsdottir, V., Masson, G., Barnard, J., Baker, A., Jonasdottir, A., Ingason, A., Gudnadottir, V., Desnica, N., Hicks, A., Gylfason, A., Gudbjartsson, D., Jonsdottir, G., Sainz, J., Agnarsson, K., Birgisdottir, B., Ghosh, S., Olafsdottir, A., Cazier, J., Kristjansson, K., Frigge, M., Thorgeirsson, T., Gulcher, J., Kong, A., and Stefansson, K., 2005: A common inversion under selection in Europeans. *Nat Genet*, **37**(2), 129–137. doi:10.1038/ng1508.

- Stephens, M., and Donnelly, P., 2003: A comparison of bayesian methods for haplotype reconstruction from population genotype data. *Am J Hum Genet*, **73**(5), 1162–1169.
- Stephens, M., and Scheet, P., 2005: Accounting for decay of linkage disequilibrium in haplotype inference and missing-data imputation. *Am J Hum Genet*, **76**(3), 449–462. doi:10.1086/428594.
- Stephens, M., Smith, N. J., and Donnelly, P., 2001: A new statistical method for haplotype reconstruction from population data. *American Journal of Human Genetics*, **68**, 978–989.
- Stoer, M., and Wagner, F., 1994: A simple min cut algorithm. In *ESA*, 141–147.
- Sturtevant, A. H., 1921: A case of rearrangement of genes in drosophila. *Proc Natl Acad Sci U S A*, **7**(8), 235–237.
- Sugawara, H., Harada, N., Ida, T., Ishida, T., Ledbetter, D., Yoshiura, K., Ohta, T., Kishino, T., Niikawa, N., and Matsumoto, N., 2003: Complex low-copy repeats associated with a common polymorphic inversion at human chromosome 8p23. *Genomics*, **82**(2), 238–244.
- Szamalek, J. M., Cooper, D. N., Schempp, W., Minich, P., Kohn, M., Hoegel, J., Goidts, V., Hameister, H., and Kehrer-Sawatzki, H., 2006: Polymorphic micro-inversions contribute to the genomic variability of humans and chimpanzees. *Hum Genet*, **119**(1-2), 103–112. doi:10.1007/s00439-005-0117-6.
- Tagawa, H., Miura, I., Suzuki, R., Suzuki, H., Hosokawa, Y., and Seto, M., 2002: Molecular cytogenetic analysis of the breakpoint region at 6q21-22 in T-cell lymphoma/leukemia cell lines. *Genes Chromosomes Cancer*, **34**(2), 175–185.
- Templeton, A., Clark, A., Weiss, K., Nickerson, D., Boerwinkle, E., and Sing, C., 2000: Recombinational and mutational hotspots within the human lipoprotein lipase gene. *American Journal of Human Genetics*, **66**, 69–83.
- The International HapMap Consortium, 2003: The International HapMap Project. *Nature*, **426**(6968), 789–796. doi:10.1038/nature02168.
- The International HapMap Consortium, 2005: A haplotype map of the human genome. *Nature*, **437**(7063), 1299–1320.
- Turner, D. J., Shendure, J., Porreca, G., Church, G., Green, P., Tyler-Smith, C., and Hurles, M. E., 2006: Assaying chromosomal inversions by single-molecule haplotyping. *Nat Methods*, **3**(6), 439–445. doi:10.1038/nmeth881.
- Tuzun, E., Sharp, A., Bailey, J., Kaul, R., Morrison, V., Pertz, L., Haugen, E., Hayden, H., Albertson, D., Pinkel, D., Olson, M., and Eichler, E., 2005: Fine-scale structural variation of the human genome. *Nat Genet*, **37**(7), 727–732. doi:10.1038/ng1562.

- Venter, G., et al., 2001: The sequence of the human genome. *Science*, **291**, 1304–1351.
- Vinson, J. P., Jaffe, D. B., O’Neill, K., Karlsson, E. K., Stange-Thomann, N., Anderson, S., Mesirov, J. P., Satoh, N., Satou, Y., Nusbaum, C., Birren, B., Galagan, J. E., and Lander, E. S., 2005: Assembly of polymorphic genomes: algorithms and application to *Ciona savignyi*. *Genome Res*, **15**(8), 1127–1135. doi:10.1101/gr.3722605.
- Wall, J. D., 2000: A comparison of estimators of the population recombination rate. *Mol Biol Evol*, **17**(1), 156–163.
- Wang, L., Zhang, K., and Zhang, L., 2001a: Perfect phylogenetic networks with recombination. *Journal of Computational Biology*, **8**(1), 69–78.
- Wang, L., Zhang, K., and Zhang, L., 2001b: Perfect phylogenetic networks with recombination. *J Comput Biol*, **8**(1), 69–78. doi:10.1089/106652701300099119.
- Winckler, W., Myers, S. R., Richter, D. J., Onofrio, R. C., McDonald, G. J., Bontrop, R. E., McVean, G. A., Gabriel, S. B., Reich, D., Donnelly, P., and Altshuler, D., 2005: Comparison of fine-scale recombination rates in humans and chimpanzees. *Science*, **308**(5718), 107–111. doi:10.1126/science.1105322.
- Wiuf, C., 2004: Inference on recombination and block structure using unphased data. *Genetics*, **166**(1), 537–545.
- Yue, Y., Stout, K., Grossmann, B., Zechner, U., Brinckmann, A., White, C., Pilz, D. T., and Haaf, T., 2006: Disruption of TCBA1 associated with a de novo t(1;6)(q32.2;q22.3) presenting in a child with developmental delay and recurrent infections. *J Med Genet*, **43**(2), 143–147.
- Zaitlen, N., Kang, H., Eskin, E., and Halperin, E., 2007: Leveraging the HapMap correlation structure in association studies. *Am. J. Hum. Genet.*, **80**, 683–691.