Algorithms for Human Genetics

by

Bonnie Beth Kirkpatrick

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

and the Designated Emphasis

in

Computational and Genomic Biology

in the

Graduate Division

of the

University of California, Berkeley

Spring 2011

Algorithms for Human Genetics

Abstract

Algorithms for Human Genetics

by

Bonnie Beth Kirkpatrick

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Richard M. Karp, Chair

Whereas Mendel used breeding experiments and painstakingly counted peas, modern biology increasingly requires computational tools. In the late 1800's probability and experimental genetics were the critical tools for discovering the gene. Today, the combined use of statistical and computational methods to make genetic and genomic discoveries has increased after the discovery of the DNA double-helix and the development of sequencing methods. By examining relationships among individuals using computational tools, geneticists have been able to understand the biological mechanisms that produce genetic diversity, map ancestral movements of populations, reconstruct ancestral genomes, and identify relatives. Furthermore, models in genetics have inspired advances in computer science, notably the model for inheritance in families is an early example of a graphical model and helped inspire the sum-product algorithm.

The genetic data of interest is single-nucleotide polymorphism (SNP) data, which are positions in the genome known to have nucleotide variation across the population. Humans are diploid individuals having two copies of each chromosome. Data for an individual can come in two forms, either haplotypes or genotypes. The haplotypes are two strings, each giving the sequence of nucleotides that appear together on the same chromosome. The genotypes, for each position in the genome, give an unordered set of nucleotides that appear. In particular the genotype is said to be 'unphased' due to the lack of information about which nucleotide appears on which chromosome.

In human genetics there are two main ways to model relatedness: evolutionary relationships between people and closer, family relationships. Evolutionary relationships, from the domain of population genetics, occur through a distant relative and leave small traces of the relationship in the genome. Family relationships are typically much closer and leave much larger traces in the genome. This thesis examines algorithms for both types of relationships.

For evolutionarily related individuals, this thesis presents the perfect phylogeny and coalescent and then examines two related questions. The first is related to privacy of genetic data used for research purposes. In order to share data from studies while hopefully maintaining the privacy of study participants, geneticists have released the summary statistics of the data. A natural question, whether individuals can be detected in the summary data, is answered in the affirmative by using a perfect phylogeny model. The second question is

how to construct perfect phylogenies from haplotypes where there is missing data. We introduce a polynomial-time algorithm for enumerating such phylogenies. This algorithm can be used to compute the probability of the data as an expectation over possible coalescent genealogies.

Recent relationships are modeled using a family tree, or pedigree graph. Traditionally, geneticists construct these graphs from genealogical records in a very tedious process of examining birth, death, and marriage records. Invariably mistakes are made due to poor record keeping or incorrect paternity information. As an alternative to manual methods, this thesis addresses the problem of automatically constructing pedigree graphs from genetic data.

The most obvious way to reconstruct pedigrees from genetic data is to use a structured machine learning approach, similar to phylogenetic reconstruction. That method would involve a search over the space of pedigree graphs where the objective is to find the pedigree graph with the highest likelihood of generating the observed data. Unfortunately, this is not a good way to proceed for two reasons: the space of pedigree graphs is exponential, and the likelihood calculation has exponential running time. The likelihood calculation given genotype data is known to be NP-hard. In an attempt to make use of the likelihood in complex pedigrees, the method PhyloPed uses a Gibbs sampler to infer haplotypes from genotype data. In a second attempt to use likelihood methods, this time for haplotype data, an NP-hardness result is presented. A third attempt to find an efficient algorithm for the likelihood problem results in a state-space reduction method for the pedigree hidden Markov model.

Since likelihood-based approaches seem completely infeasible, a completely different approach is introduced. We focus on the problem of inferring relationships between a set of living individuals with available identity-by-descent data. For convenience, we assume that the inferred pedigree is monogamous without inter-generational mating. Two heuristic and practical pedigree reconstruction methods are introduced, one for inbred pedigrees and the other for outbred pedigrees. This work immediately reveals another important problem, that of evaluating the resulting inferred pedigree against a ground-truth pedigree. This can be done either by determining whether the two pedigrees are isomorphic or by finding the edit distance between the two pedigrees.

To my parents, George and Denise Kirkpatrick.

"A man should learn to detect and watch that gleam of light which flashes across his mind from within, more than the lustre of the firmament of bards and sages. Yet he dismisses without notice his thought, because it is his."

"There is a time in every man's education when he arrives at the conviction that envy is ignorance; that imitation is suicide; that he must take himself for better for worse as his portion."

–Ralph Waldo Emerson, *Self-Reliance*

Contents

List of Figures

Acknowledgments

Many wonderful people have contributed to my academic pursuits. Chiefly among them are Richard Karp and Eran Halperin who together coached me through graduate school. Professor Karp, thank you for being very approachable and for constantly asking me to find simple and clear ways to communicate our work. Professor Halperin, thanks for always sharing career advice and for always encouraging me to work on very practical problems. Thanks also go to Yun Song and Michael Jordan whose appreciation of statistics has been a pervasive influence on my work. Perhaps the most formative influence of all, my sister, Kay Kirkpatrick, thank you for your early insistence that I take math courses and your continued mentoring which has shaped my academic pursuits through college, graduate school, and beyond.

Before arriving at Berkeley, I had a number of outstanding mentors. Gwen Jacobs, thank you for encouraging me to study computational biology. Binhai Zhu, thanks for teaching me algorithms and complexity theory and more recently collaborating with me on some of the work that appears in this thesis. Brendan Mumey, thanks for supervising my undergraduate research at Montana State University. Nancy Amato, after mentoring me through two summers of research at Texas A & M University, deserves many thanks for suggesting that I would enjoy graduate school.

Thanks go to all my co-authors who made each research project worth doing. Every one of you made valuable contributions to the research and to my graduate education. Without you the projects appearing in this thesis would not have gotten finished.

Thanks to the Designated Emphasis for providing a venue for a computational biology community at Berkeley. Thanks to Samantha Riesenfeld, Sriram Sankararaman, Meromit Schuster, Ma'ayan Bresler, and Josh Paul for building a computational biology community in Computer Science. Thanks to Anna-Sapfo Malaspinas, I will never forget the Judo class we took as a break from all the research.

Without my dear chums, graduate school would have been a lot less pleasant. Ben and Juliet Rubinstein, Alex Simma, Louis Alarcon, Zach Anderson, Leon Barrett, Pratik Patel and Anupama Bowonder, Todd and Cheryl Templeton, Alan and Jessica Wu, and Subbu Venkatraman and Praveena Garimella, without you all, I would not have survived the rigors of graduate school. Thank you for all the Friday night dinners, Settler's games, barbecues, and holiday parties. I will miss you all more than words can express. Thanks also to my dear friends Roger and Olga Pearce. We have struggled through graduate school together, even though at different schools.

Finally, to all my family, I owe a debt of gratitude for all the support and love that has kept me grounded throughout graduate school. My parents George and Denise, thanks for nurturing inquisitive minds in both your children. Kay Kirkpatrick and Pierre Albin, my sister and brother-in-law, thanks for always giving me the benefit of your support and advice. Lauren Barth-Cohen, thanks for continuing to believe in me.

Chapter 1

Introduction to Human Genetics

Genetics is fundamental to understanding biology. The instructions for the cell are primarily encoded in DNA which is inherited from parent to child. This chapter introduces the biological processes of inheritance and mutation, as well as the motivating questions behind computational genetics research. Some of this introductory material was taken from [57].

1.1 Genetics and Inheritance

The genome, in the form deoxyribonucleic acid (DNA), is believed to encode much of the information for the development and function of living organisms. Over the last decade, large quantities of human genomic data have become available. These data specifically identify genetic variation between people. In order to understand genetic variation, we first need to consider the biological medium, DNA molecules, and how they encode variation. We also need to understand inheritance, the origin of similarity between people, and its relationship to DNA. Both variations and similarities appear in most modern data sets, because the locations of the variants are sampled with sufficient density along the DNA sequence. As a result, these biological observations provide the intuition behind the algorithms we develop in later chapters.

1.1.1 Genetic Variation

DNA and Chromosomes. A strand of DNA is a sequence of repeating units, named *nucleotides*, together with the phosphate and sugar groups which bond covalently to provide the connected backbone structure between the nucleotides. The four possible nucleotides are represented with the letters: A, C, G, and T. Each nucleotide can form a hydrogen bond with the complementing nucleotide on another DNA strand: usually A bonds with T, and G with C. DNA is energetically stable when found as a double helix containing two hydrogen-bonded and complementary strands of DNA. One long DNA double-helix is coiled, with the aid of histone proteins, into a structure named a *chromatid*. A *chromosome* is an X-shaped structure formed by joining two identical chromatids. Each chromosome contains a 4-fold redundancy of the genetic information, since there are two identical chromatids each containing a double helix which is formed from a DNA strand and its complement.

Humans have 23 types of chromosomes, numbered 1-22 and the 23rd being the sex chromosome. Each type of chromosome encodes a different set of *genes*, with each gene being the blueprint for a functional unit (protein or non-coding RNA). Both the unique physical features of a chromosome and the unique set of genes it contains identify a chromosome as a specific one of the 23 chromosome types. The 23rd chromosome is the sex chromosome, and it comes in two varieties, either an X or a Y. For the remainder of this discussion, we will leave aside the complexities of the sex chromosomes and will restrict our discussion to the 22 autosomal chromosomes. Most human cells are diploid, meaning that they contain two 'copies' of each chromosome type. Thus, a diploid human cell contains 46 chromosomes. It is important to note that the two copies of each chromosome, although identical in type, are *not* identical in genetic content. To emphasize this difference, we refer to these copies as *homologous chromosomes*.

An analogy from computer science that may be helpful is that of classes and instantiated objects. One can think of the 22 autosomal chromosomes as classes which each contain different variable declarations, or different sequences of nucleotides. Having diploid human genetic material is similar to having two instantiations of each of the chromosome objects, and the exact nucleotide content is slightly variable across instantiations of the same chromosome.

Single-Nucleotide Polymorphisms. A location in the nucleotide sequence where variation occurs is named a *polymorphic site* or *locus*. The variants appearing in the genome at a polymorphic location are referred to as *alleles*. On average, 99.9% of sites in the human genome are identical in all humans [93]. Much genetic variation occurs at single-nucleotide locations, and these sites are known as *single-nucleotide polymorphisms (SNPs)*. Thus, in the case of SNPs, the alleles are nucleotides.

Continuing with the object analogy introduced above, we can think of the variable content of a single chromosome class as a long array of nucleotide elements. Notice that the array only contains the information on a single piece of DNA from each chromosome, since there is no need to represent the 4-fold redundancies. Each gene is then described as a tuple of indices which give the the start and end positions of the gene in the nucleotide array. For the purposes of the analogy, we can think of a SNP site as a single polymorphic position in the nucleotide array. In the interest of accuracy, we note that although the term 'position' is useful for conceptual explanations, it is not synonymous with 'site' or 'locus' since the latter two terms implicitly account for variations in sequence length across individuals.

The alleles of each SNP would be the set of nucleotides that can possibly appear in the sequence at the SNP locus. In practice, the alleles of a particular SNP are determined by examining many genomes for variation at that particular site. While SNPs are the genetic variants on which we focus in this thesis, there are other types of variation that are amenable to the methods presented in this thesis, such as microsatellite loci, copy-number variants, and some structural variation.

1.1.2 Meiotic Inheritance

Since most sites in the genome are identical across all humans, there are some significant sources of genetic similarity. These sources are explained by the process of inheritance which transmits genetic material from parents to their offspring. In diploid organisms with sexual reproduction, each parent contributes half of their genetic material to each offspring. One of the offspring's homologous chromosomes comes from the father, and the other is contributed by the mother. A *haplotype* is a sequence of alleles that were inherited together from a single parent and all appear on the same chromosome in an individual.

This simple picture of direct inheritance has two additional processes that produce haplotype diversity: recombination and mutation. Although these two mechanisms are most clearly observed in families, the effects of recombination and mutation are also seen among groups of 'unrelated' humans due to the evolutionary relatedness of the human species.

Recombination. Meiosis is the form of cellular reproduction that produces gametes, eggs and sperm, for sexual reproduction. A *recombination* event occurs during meiosis in the parent just prior to gamete or zygote formation and results in DNA being swapped between the two homologous parental chromosomes. This forms a new combination of alleles in the haplotype being transmitted to the offspring. Thus, the genetic material contributed by a single parent is from both grandparents, because recombination yields a recombinant chromosome containing portions of both grand-parental chromosomes (see Fig. 1.1). For more molecular details about recombination, refer to Hartwell et al [38]. Between one and three recombination events occur per chromosome per generation. The average rate of recombination between contiguous nucleotides is 10^{-8}, but recombination rates vary locally by at least four orders of magnitude [68]

Recombination is usually modeled by the probability θ of a recombination event between two loci. For instance consider two SNPs, C_1 and C_2 with alleles $\{A, T\}$ and $\{G, C\}$ respectively. In a parent having haplotypes AG and TC, the *recombinant* gametes have haplotypes AC or TG while the *parental-type* gametes have AG or TC. The farther apart two loci appear on the DNA sequence, the larger the probability of observing recombinant gametes. As a result, loci that are very far apart are said to be *unlinked* with $\theta \simeq 0.5$. If two loci are contiguous, or nearly contiguous, in the genome, they are said to be *linked* because the vast majority of the gametes carry the parental-type alleles ($\theta << 0.5$). Although the frequency of recombinant gametes is usually only counted in a family study or in sperm study, there is a related concept that describes the historical effects of recombination in a largely-unrelated population. *Linkage disequilibrium (LD)* measures linkage by examining the haplotypes in the population and determining whether the alleles appearing at two loci are correlated. There are several different statistics for LD (see [4] for details).

Mutation. Rather than producing new combinations of existing alleles, a *mutation* is an event that introduces a new allele. The mutation rate per SNP is estimated as being between 10^{-8} and 10^{-9} [99]. One formulation of mutation that is useful at the population-level is the perfect phylogeny. The main assumption underlying it is the infinite-sites assumption,

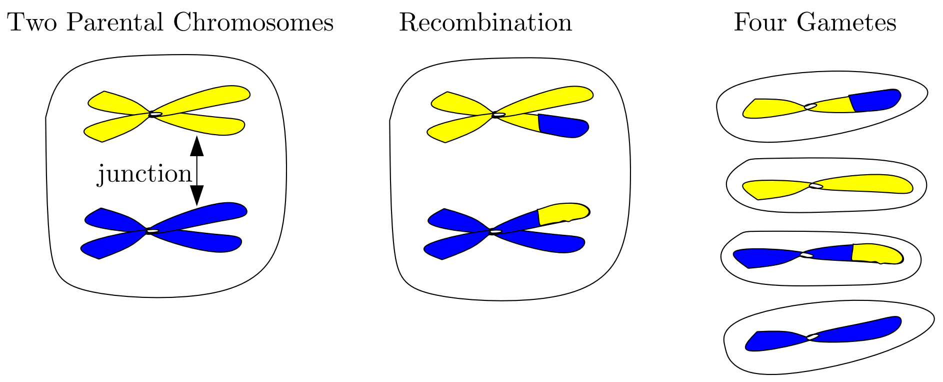


Figure 1.1: **Recombination.** This figure illustrates the two parental chromosomes and four gametes that result from the recombination event at the indicated recombination junction on the parental chromosomes. The parental chromosomes are homologous and distinguished by their colors. The chromosomes that appear together in one cell are encircled by a line. The two recombinants are a collage of the parental chromosomes. Only two of the four resulting gametes contain recombinant chromosomes, which are colored with both yellow and blue.

where each site can only mutate once over history. Of course, if there really are countably infinite sites, the genome-wide mutation rate is spread across so many sites that there is little chance of mutating the same site twice.

Among unrelated people, inheritance and evolution have resulted in a small number of short haplotypes being shared by many people. If a haplotype is untouched by either recombination or mutation, it will propagate through a phylogeny and will be shared by many individuals. Recombination would seem to diversify away the effects of inheritance by re-assorting the haplotypes, but linkage results in short regions of the genome having conserved haplotypes. The process of mutation also diversifies haplotypes, but at a sufficiently low rate that local haplotypes will differ in only a few positions. These ideas inspire the perfect phylogeny method introduced in Chapter 2.

1.1.3 Data

Perfect data would tell us both the haplotype of each individual and exactly where the recombination breakpoints occurred, see Figure 1.2. Currently, sperm typing and family studies are the only way to reliably determine where recombination breakpoints occur. In sperm studies, the haploid gametes in sperm are genotyped to determine where recombination breakpoints are[18]. In family studies, the recombinations must be inferred from genotype data. There is some hope in the near future of obtaining haplotype information via sequencing methods, although this data is currently unavailable.

Genotyping Although haplotypes are the genetic variants that are most useful for genetic studies, using laboratory methods to determine haplotypes from diploid cell samples is

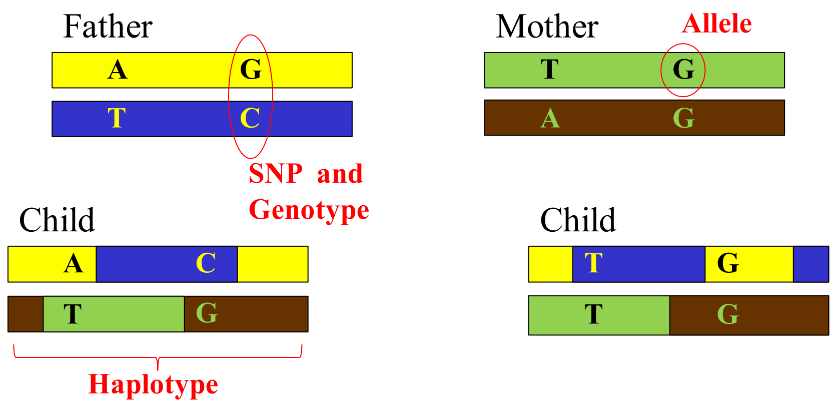


Figure 1.2: **Perfect Data.** One chromosome is illustrated with two copies for each person (rectangles with the sequence of the genome being along on the x-axis). The colors represent which regions of the chromosomes are identical to those in the parent. Each child receives a collage of their parents two chromosomes, with one collage coming from each parent. The positions in the genome where the colors change represent recombination breakpoints. We would like to assay the color pattern, since this would represent complete data. However, there is no experiment which can tell us exactly where recombination breakpoints are. Instead, we typically use genotyping to assay particular positions in the genome, called single nucleotide polymorphisms (SNPs), to determine what nucleotide alleles appear there. Two SNP positions are illustrated here. There is some hope that in the near future haplotype data may become available from next-generation sequencing technologies.

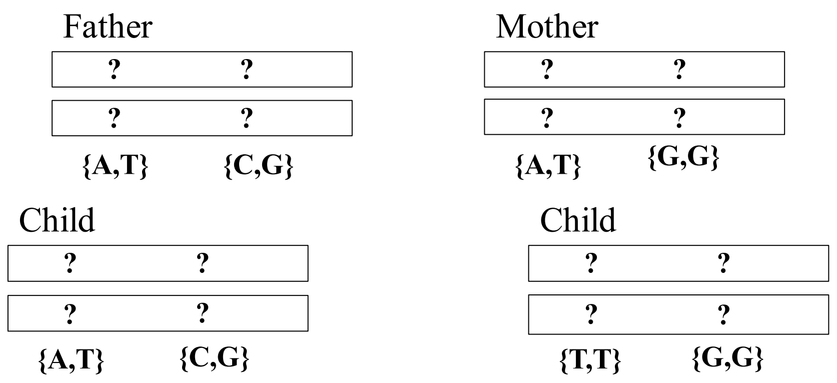


Figure 1.3: **Genotype Data.** The same chromosomes are shown here as in Figure 1.2. As previously mentioned, we would really like to know exactly where the recombination breakpoints occur. However, genotyping only tells us the unordered set of alleles that appear at each site. Two sites are shown here. Both the haplotype information and the recombination breakpoints must be inferred from the genotype data.

currently prohibitively expensive for large numbers of individual. Instead most studies perform genotyping, an affordable analysis of genetic variation that is performed on diploid cell samples. A genotype experiment examines SNPs at particular loci in the nucleotide sequence and determines which alleles appear in the pair of homologous chromosomes.

The *genotype* of an individual reveals the (unordered) set of alleles that appear at each site. Figure 3.7 illustrates the genotypes of a portion of the DNA sequence belonging to a diploid individual. A single locus can contain two distinct alleles, in which case it is *heterozygous*, or one allele, in which case it is *homozygous*. In the mother in Figure 3.7, the second locus is homozygous, while the other is heterozygous. Notice that the genotype is symmetric, since the set of alleles revealed by the analysis does not contain any order information. For individual i and locus j, we will use the notation

$$\{g_{ij}^0, g_{ij}^1\}$$

to refer to the individual's two alleles.

The order information is contained in the haplotypes of an individual. In Figure 1.2, the father has two haplotypes AG and TC. There are two haplotype pairs that satisfy the father's genotypes shown in Figure 1.3. In general, when there are H heterozygous sites in the observed genotype, there are 2^{H-1} haplotype pairs that could have produced the genotype. We will represent the haplotypes as a vector of alleles. So, individual i will have two haplotype vectors

$$(h_i^0, h_i^1)$$

with h_{ij}^k being the allele at site j. Genotypes and haplotypes for the same person must be *consistent*, so that the set of alleles at each site are identical

$$\{g_{ij}^0, g_{ij}^1\} = \{h_{ij}^0, h_{ij}^1\}.$$

Since genotyping is inexpensive relative to haplotyping, most studies based on haplotypes collect genotype data and infer the haplotypes from the genotypes (also called the *phase* information). Due to the symmetry properties relating genotypes to haplotypes, the decision to collect genotype data introduces a non-trivial problem of inferring the haplotypes.

Sequencing Whole genome sequencing methods are currently being developed. These methods will allow us to examine genetic variation at an unprecedented resolution. Not only will we be able to detect unknown rare variants and examine structural variation (such as chromosomal rearrangements), but using families we will be able to observe distinct mutation events that produce novel variants. Exome sequencing is one method producing invaluable information about novel variants [39].

Single-molecule sequencing is another genome sequencing technology where the genome is sheared into many small fragments and then the fragments are sequenced by using DNA replication machinery to bind complementary florescent nucleotides to the sheared fragments of DNA. A camera is used to read the colors of the florescent nucleotides as they are consecutively attached to the fragments yielding a sequence "read" of the fragment. Single-molecule sequencing is an attractive alternative to genotyping and may soon yield long haplotype reads for individuals [26]. Such technologies are being developed and may become commercial within five to ten years. Sequencing methods would yield more information from the same set of sampled individuals than genotyping methods, because more of the phase information would be known.

1.2 Motivating Questions

The driver behind computational genetics research is a desire to fully understand genetic epidemiology, pharmacogenomics, cancer genetics, population histories, and inheritance processes. All of these goals require a sophisticated understanding of inheritance and the functional expression of each unique genome.

The genome encodes most of the information required to orchestrate cellular activities. Genes are expressed in the form of proteins or non-coding RNAs to carry out cellular functions. Cascades of chemical reactions involving multiple proteins and RNAs lead to complex cellular processes. Measurable characteristics of these processes are called *phenotypes*, or an observable trait of an organism. The grand challenge of genetics is go understand how the genome encodes phenotypic variation, and how small changes in the genome can result in varying disease susceptibility.

A complete understanding of genetics, genetic variation, and the functional consequences of genetic variation will greatly aid in understanding disease and treatment response. Every individual has their unique genome and their unique disease susceptibilities. Furthermore, every individual has a unique response to drug treatment that is dependent on their unique genome through their ability to metabolize drugs. The goal of genetic epidemiology is to elucidate the relationship between the genome and disease. Pharmacogenomics is aimed at understanding how genetic variation influences drug response in an individual manner.

Much of the work in both genetic epidemiology and pharmacogenomics involves attempts to correlate the presence or absence of a genetic variation with a disease or a drug response. In these studies, researchers collect genetic material from individuals who have a particular disease, called cases, and from individuals without the disease, called controls. In studies with unrelated individuals, association studies are done which are statistical tests for correlation. These are often regression tests with allelic variations being the independent variables and the disease being the dependent variable [77]. If there is a correlation detected, then the tested variation is presumed to be involved in production of the disease phenotype. Of course, biological investigation must follow-up the statistical study to determine whether the detected variation is actually linked to a gene that might be causative for the disease. These studies have successfully found genetic variations involved in Celiac disease, diabetes, Crohn's disease, and heart disease.

The classical association study is done with unrelated individuals. There is a similar study, known as linkage analysis, which is performed on families[72]. Presuming that there is a single site of variation responsible for the disease, the goal is to find the location in the genome that best explains that site. The principal feature of the data that is exploited is linkage disequilibrium, the correlation between neighboring sites in the genome. These studies have found the genetic basis of many diseases including cystic fibrosis, Huntington's, and sickle-cell anemia.

Another critical application of computational methods is cancer genetics. The somatic genome of cancer cells carries mutations that produce malignant behavior from the cancer cells [89]. The genomes of cancer cells and normal cells differ in critical places that influence the production of the cancerous phenotype. Since mitosis proceeds without recombination (in contrast to meiosis), somatic genomes typically reproduce with few changes. Cancer is usually the product of particular mutations occurring in genes whose disruption can produce a cancerous phenotype.

It is also becoming evident that a population genetic perspective is required to understand the disease phenotype of virus populations. It seems that the infectious properties of HIV require that a population of the virus be present in the infected person's blood. The existence of multiple virus genomes (i.e. with small changes from the ancestral virus genome) allow the virus population to be more robust to environmental changes and influences the virulence of the virus population [60].

Population genetic problems involve inferring the history of human populations, such as bottlenecks and divergences. Very recent investigations involve the question of whether Neanderthals mated with humans and how recently [33]. Many forensic questions are also of a population genetic nature, such as identifying the owner of a tissue sample left at a crime scene [86]. Finally there are important questions of genetic privacy that are becoming more relevant as genotyping technologies become more commonplace.

Family-based analysis are useful beyond just finding disease-gene correlations using linkage analysis or association testing. Genotypes of family members can be used to observe evidence of novel variants, those variants not present in parents but appearing in their children. Novel variants are mutations and provide a way to investigate mutation rates, particularly using whole genome sequencing methods which are more likely than genotyping methods to capture the presences of rare mutations. Fine-scale recombination rates can be mapped using family studies. Families provide the only known way to estimate recombination rates in human females [20]. On the other hand, male recombination rates can be inexpensively estimated from sperm-typing studies.

Additionally there are a number of questions regarding inferring relationships. Paternity testing is one of the most common relationship testing methods. Beyond that, one might test for any particular relationship between a pair of individuals. Even more important is the problem of inferring relationships on a set of individuals.

1.3 Computational and Statistical Challenges

In this thesis, we will focus on a number of specific computational and statistical challenges. The problems addressed here include privacy of individuals participating in genetic studies, pedigree analysis with haplotype data, inferring haplotypes in large pedigrees, inferring recombination breakpoints from haplotype data, and inferring pedigrees from genotype data.

There are a number of interesting population genetic questions. In Chapter 2 we will focus on two questions. The first is the privacy of individuals participating in genetic studies. It was originally thought that average allele frequencies across the individuals in a study could be released without violating the privacy of study participants. This idea was strikingly false [41]. Indeed, we will investigate how haplotype information can be used to more accurately detect individuals in studies than genotype information. The second population genetic question in this thesis is that of the compatibility of partial binary characters with the perfect phylogeny. In general this problem is NP-hard, however we consider a particular variety of data under which this problem is solvable in polynomial time. In particular, we can enumerate the perfect phylogenies that are compatible with a given set of partial characters. This has applications to computing the probability of data under the coalescent with infinite sites. This is the first known algorithm for these calculations on missing data.

Next, we consider questions regarding family genetics. In Chapter 3, after introducing family trees, or pedigrees, we introduce hardness results for pedigree calculations given haplotype data. Regardless of the hardness of these calculations, we proceed to introduce three different algorithms for pedigree calculations in Chapter 4. The first algorithm infers haplotypes for all individuals in a large pedigree on a small number of loci. Haplotype inference of this type can improve the power of association studies on pedigrees. The second algorithm infers recombination breakpoints from haplotype data. It turns out that when there is missing data for some of the individuals in the pedigree, haplotype data and genotype data can be equally useful for inferring recombination breakpoints. Finally, we introduce a method for improving the efficiency of pedigree likelihood calculations.

In Chapter 5 we consider methods for pedigree reconstruction. First, we introduce a theoretical method for pedigree reconstruction that has conceptual value. Next, we introduce the problem of evaluating the accuracy of inferred pedigrees. This problem can be formulated either as a pedigree isomorphism problem or as an edit distance problem. The edit distance problem turns out to be APX-hard, however there appear to be efficient and useful heuristics for approximating the edit distance. Finally, we introduce two practical methods for inferring pedigrees from identity-by-descent data. The simulation results indicate that these methods work better than the state-of-the-art methods. We then discuss an application of this method to publicly available data.

The final contribution of this thesis is to discuss future problems in Chapter 6. Several natural and important problems are proposed. In particular sequence data is opening new opportunities to modify existing algorithms and make genetic discoveries. In addition, the pedigree reconstruction problem is not solved. There is much work left to do both in evaluating pedigree reconstruction methods and in improving the methods.

Chapter 2

Unrelated Individuals

Strictly speaking no set of individuals is 'unrelated', since even individuals of different species are related phylogenetically. To clarify the context of this chapter, we are interested in individuals who are not recently related, i.e. not related via close family relationships. In particular, we imagine a population of haploid individuals who mate randomly with each other (i.e. no biased selection of mates) and are not subject to the forces of selection. The main process is mutation and genetic sequences are generated in the model via the introduction of mutated alleles. This random process is the *coalescent process.*

Why would we want to model a diploid human population using a haploid model with so many restrictions? The reason is that this model can be dealt with mathematically, in that we can actually compute certain quantities of interest. It turns out that the genealogies generated by the coalescent process are trees with randomly chosen branch lengths. As we know, trees are convenient both for closed-form solutions for quantities of interest and for efficient computation. Under more complicated, non-tree models, such as the coalescent with recombination, the genealogies, called *ancestral recombination graphs (ARGs)*, are more difficult to perform calculations with [40].

Under one particular model of mutation, the coalescent with infinite sites, the genealogies generated by the coalescent must be consistent with the perfect phylogeny tree [101]. For a given perfect phylogeny tree, there are many coalescent genealogies consistent with it, thus computing probabilities of data under this model requires enumerating genealogies consistent with the perfect phylogeny. Fortunately, the perfect phylogeny can be found efficiently for binary data when there is no missing data [36].

This chapter introduces the coalescent with infinite sites and the perfect phylogeny. Then it proceeds to introduce two algorithms. The first algorithm is an application of the perfect phylogeny to the problem of detecting individuals in pooled data sets. Geneticist have shared genotype data from studies by pooling the individuals in the study and releasing the average allele frequencies for each site. Recently this pooled data was shown to violate the privacy of individuals participating in the study, since individuals can be detected in the pooled data [81]. The algorithm introduced in this chapter extends previous work by leveraging haplotypes to better detect individuals in pooled data.

The second algorithm introduced in this chapter considers efficient ways for constructing perfect phylogenies from binary data with missing values. While this problem is hard in general, there are settings in which the phylogenies can be found in polynomial time. In particular, we introduce an enumeration algorithm which finds all the perfect phylogenies consistent with the input data, provided that the input data satisfy a certain condition. This algorithm gives the first known way to compute probabilities of observed data with missing values under the coalescent model with infinite sites.

2.1 Populations

The 'population' referred to in the title 'population genetics' is one that satisfies the assumptions of the coalescent process. These are populations that are panmictic, or well mixed by random mating, and not subject to selection. The study of family relationships is a different topic typically falling outside the purview of population genetics. The panmictic populations referred to here are typically populations of haploid individuals that evolve by a mutational process without recombination. Although more recent models have considered recombination and diploid individuals [40].

One of the oldest population models was introduced by Wright [100] and Fisher [30] in 1931 and 1930, respectively. This was after the discovery of the gene and Mendelian genetics, but before Watson and Crick discovered the structure of DNA and long before the advent of molecular genetics. The advent of sequencing and genotyping methods changed the field of theoretical population genetics into a practical science with applications to forensics, epidemiology and pharmacogenomics.

2.1.1 Coalescent

The coalescent [47] defines a genealogical process on haploid individuals (where each individual has a single copy of each chromosome), for example bacteria or yeast. The coalescent primarily models mutation and is the limiting result for the Wright-Fisher model on an infinite population [98]. The properties of the coalescent have been beautifully developed in many books which also give the relationship between different variations on the coalescent model [98, 40]. One very well studied version of the coalescent is the coalescent with infinite sites, which is related to the perfect phylogeny [82], and is a model used both for evolutionary and population-genetic relationships. Much of the literature devoted to the coalescent deals with the problem of sampling coalescent trees from the probability distribution of the genealogy given the observed data (typically from the most recent generation). In this section, we will restrict ourselves to discussing the Wright-Fisher (WF) model and Kingman's n-coalescent with the infinite-sites mutation model without recombination. Both WF and Kingman's coalescent can be thought of as a directed tree whose nodes are haploid individuals and whose edges describe inheritance from an individual in an older generation to one in a more recent generation.

Wright-Fisher Model.

The Wright-Fisher (WF) model consists of a fixed number, N, of haploid individuals per generation and a fixed number of generations t. The generations are discrete with no mating between generations. To simulate reproduction going forward in time, each haploid individual chooses their haploid parent uniformly at random from the previous generation, i.e. each parent has chance $1/N$ of being chosen. The number of offspring of a particular individual is Poisson distributed [40]. The WF model is a coalescent process, because when starting with the most recent generation and looking backward in time, there is always a single ancestor from which all the extant individuals are descended, termed the *most recent common ancestor (MRCA)*. This means that the genealogy of relationships forms a tree. (Note that some versions of the WF model for diploid individuals do not necessarily coalesce [73].)

The WF model has discrete generations, but it is mathematically convenient to consider continuous time approximations. Let n be a fixed sample size of *extant* individuals taken from the most recent generation. When taking the limit as $N \to \infty$ the WF model converges to the continuous-time n-coalescent described by Kingman [47].

Frequently the population size is re-scaled to obtain an *effective population size* that facilitates the conversion between discrete-time units and continuous-time units. Specifically, a common scaling is such that one unit of continuous time is the average time in the WF model for two individuals to coalesce into a common ancestor, which is $2N$ generations.

The n-Coalescent

We will limit our discussion to giving the stochastic process to sample coalescent genealogies on n extant individuals. Certainly this discussion does not do justice to the breadth of work on the coalescent. Please see [40, 98] for much more in-depth expositions.

To sample a coalescent genealogy, we begin with n individuals. Let k represent the number of branches in the tree.

1. Initially, let $k = n$.
2. While $k > 1$ simulate events as follows.
   (a) The waiting time T_k^c to the next coalescent event backwards in time is exponentially distributed as $T_k^c \sim Exp(\binom{k}{2})$.
   (b) To choose the particular event, pick a random pair (i, j) of branches where $1 \leq i < j \leq k$ uniformly among the $\binom{k}{2}$ possible pairs of branches.
   (c) Merge lineages i and j into one branch updating the branch count $k = k - 1$.

The Infinite Sites Mutation Model

Here we are concerned with simulating the haplotypes of the individuals in addition to the genealogy. We restrict our attention to the infinite sites model for mutation where each new mutation occurs at a unique position in the genome and the length of the haplotype is the number of mutations. The genome is assumed to be infinitely long so that a uniform at random choice of a site to mutate always produces a unique site. This means that there are no *recurrent* mutations, i.e. mutations at the same site occurring at different time points. This is only one mutation model, please see [40, 98] for other mutation models.

We can modify the WF model to simulate haplotypes under the infinite sites model as follows. For each reproduction event, the parental haplotype chosen is passed on unmodified with probability $1-u$. With probability u the parental haplotype undergoes a mutation event at a unique site. Once mutation occurs at a site, that particular mutant allele is inherited to all the individual's descendants. Typically use a bit to represent each site where $a \in \{0,1\}$ represents the ancestral allele and $1-a$ represents the mutant allele.

We can similarly modify our generative model for sampling an n-coalescent. Let ρ represent the population mutation rate which is a scaled WF mutation rate, typically $\rho = 4Nu$. Again, let k represent the number of branches in the tree.

1. Initially, let $k = n$.

2. While $k > 1$ simulate mutation and coalescent events as follows.

   (a) The waiting time to the next event backwards in time is exponentially distributed $Exp(k(k-1+\rho)/2)$.

   (b) Choose the type of event, either coalescent or mutation, as follows. With probability $(k-1)/(k-1+\rho)$ the event is a coalescent event and with probability $\rho/(k-1+\rho)$ it is a mutation event.

   (c) To choose the particular coalescent event, pick a random pair (i,j) of branches where $1 \leq i < j \leq k$ uniformly among the $\binom{k}{2}$ possible pairs of branches, merge lineage i and j, and let $k = k-1$.

   (d) To choose a particular mutation event, pick a branch to mutate uniformly at random from the k branches. Leave the number of branches, k, unchanged.

Computing probabilities of observed data under the coalescent model with infinite sites involves effectively enumerating all the genealogies consistent with the data and determining the probability of each genealogy. This can be done reasonably effectively by enumerating the perfect phylogenies consistent with the data and then enumerating the genealogies consistent with the perfect phylogenies using dynamic programming [101]. To discuss this, we need to introduce the perfect phylogeny and algorithms for finding perfect phylogenies compatible with the data.

2.1.2 Perfect Phylogeny Tree

The perfect phylogeny tree is a variant of the coalescent. Specifically, it is the coalescent with infinite sites, but without the branching order specified by the coalescent genealogy. In other words, many coalescent genealogies are consistent with one perfect phylogeny tree, see [101] for details.

	C_1	C_2	C_3	C_4	C_5
S_1	1	0	1	1	0
S_2	1	1	0	0	1
S_3	1	0	0	1	0

Table 2.1: **Compatible characters.** Here the rows are haplotypes and the columns are SNPs. In the language of phylogeny, the columns are called *characters*. Characters are *compatible* if there exists a perfect phylogeny for the haplotypes.

A perfect phylogeny on binary characters (in this case SNPs, see Table 2.1) is defined as a rooted tree having haplotypes as nodes and SNPs as edges. Every edge in the tree, labeled with a SNP, represents a $a \to 1 - a$ mutation at that SNP where $a \in \{0, 1\}$ is the ancestral state of the SNP. Note that this tree is not necessarily a binary tree. Since there are many inheritance events that could violate a tree-like structure, we will introduce the two assumptions that guarantee the phylogeny to be a tree:

1. The number of sites is infinite relative to the genome-wide mutation rate, which allows each site to mutate at most once in the phylogeny.

2. No recombination occurs between haplotypes, and thus each haplotype has a single ancestor.

From property (2), it is easy to see that the phylogeny must be a tree, since each haplotype can only have one ancestor. The definition of the perfect phylogeny tree as rooted assumes that there is a most recent common ancestor (MRCA) for the phylogeny. Property (1) tells us that each haplotype containing a mutation descended from the ancestral haplotype where the mutation first occurred. Let each node of the tree represent a haplotype, then the root of the tree is the MRCA of the extant haplotypes. Each edge in the tree represents a mutation that occurred between a parent-node and a child-node. Parent and child relationships in the tree are indicated by two nodes sharing an edge, and the parent node has the shorter path to the root. This figurative use of parent and child should not be confused with genetic relationships such as Mendelian inheritance. Thus, every edge has two adjacent haplotypes which differ by a single SNP, and that SNP labels the edge (Figure 2.1). Property (1) tells us that there is at most one edge in the tree that corresponds to each mutation, and this edge corresponds to the one time in history when the mutation occurred. Therefore, there are no recurrent mutations.

For an example of a phylogenetic tree see Figure 2.1. There are three haplotypes on five sites. Only four of the sites mutate, and these four sites, $\{C_2, C_3, C_4, C_5\}$, label the tree branches on which they mutate.

Compatibility of Data

Wherever possible we will follow the notation used by Semple and Steel [82]. A *phylogenetic tree* $\mathcal{T}$ is an ordered pair (T, ϕ) with a label set X, where $T = (V, E)$ is a tree and $\phi : X \to V$ is called the *labeling map*.

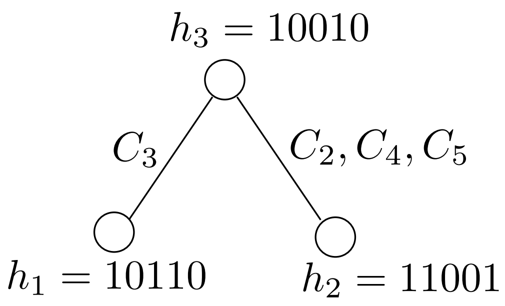


Figure 2.1: **Compatibility.** The characters given in Table 2.1 are convex on this tree. The root of the connected subtree on which a character's 1-states are convex is labeled with that character.

A *character* is a map $C : X' \to A$ from a non-empty subset of the label set to the set of character states. If $X' = X$, the map C defines a *full character.* If $|A| = 2$, the character is called a binary character and the states are labeled $A = \{0, 1\}$. A non-full character, C, having $X' \subseteq X$ and $X' \neq X$, is a *partial character.* For example, Table 2.1 is a data set where the rows of the table are haplotypes. The columns are characters, which are simply the values of all the haplotypes at a particular site. The characters can be thought of as SNPs. Here the data are for full characters since there is no missing data.

A set of *partial characters* $\mathcal{C} = \{C_1, C_2, ..., C_m\}$ has some unspecified character states on X where X is the label set obtained from the union of the domains of C_i for all i. A *resolution* of $\mathcal{C}$ is a full character set $\mathcal{C}^R$ which agrees with $\mathcal{C}$ in every specified character state and gives some assignment for the unresolved characters.

A character $C : X \to A \cup \{*\}$ is *convex* on a phylogenetic tree $\mathcal{T} = (T = (V, E), \phi)$ if and only if there is an extension $\bar{C} : V \to A$ of the character such that the following conditions hold:

i) The restriction of the domain of $\bar{C}$ to X' yields a function equivalent to C.

ii) Let T_a be the subgraph of T induced by the set of vertices mapped by $\bar{C}$ to a particular character state a. For every character state $a \in A$, T_a is connected and disjoint from all other T_b for $b \in A$, $a \neq b$.

A set of characters is *compatible* if and only if there is a phylogenetic tree on which all the characters are convex. A set of partial characters $\mathcal{C}$ is *compatible* when there is a fully-specified resolution $\mathcal{C}^R$ that is compatible.

In general there are three distinct problems pertaining to compatibility of characters.

i) Given a phylogenetic tree $\mathcal{T}$ and a set of characters, determine whether the characters are compatible with the tree.

ii) Given a set of characters, determine whether the characters are compatible and construct a tree on which the characters are convex.

iii) Given a set of characters, find the maximal subset of characters that are compatible.

Problem (i) is easily computed in linear time. Problem (ii) is known as both the character compatibility problem and the perfect phylogeny problem. In the general case, both problems (ii) and (iii) are NP-hard [82]. For the case of binary characters, there is a polynomial algorithm for determining compatibility [36]. Even for binary partial characters, problem (ii) is NP-hard [88].

For example the characters given in Table 2.1 are compatible with the phylogenetic tree in Figure 2.1. In this case, there is only one phylogenetic tree compatible with the data, since the characters are full binary characters.

Perfect Phylogeny Haplotyping

The perfect phylogeny compatibility problem described above is defined for haplotype data. Due to the ubiquity of genotype data, it is useful to define a notion of compatibility between genotype data and a perfect phylogeny tree. Let g_{ij}^0 and g_{ij}^1 be the two binary genotype alleles for individual i at site j, with $g_{ij}^k \in \{0,1\}$. The genotype of person i is *compatible* with a perfect phylogeny tree if there exist two tree haplotypes h^0 and h^1 with binary alleles such that $g_{ij}^0 + g_{ij}^1 = h_j^0 + h_j^1$ for all sites j. A set of genotypes are compatible with a perfect phylogeny tree if all the genotypes in the set are compatible with the perfect phylogeny tree. This problem is known as the *Perfect Phylogeny Haplotyping* problem.

The problem of finding perfect phylogenies compatible with genotype data was first proposed by Gusfield [35]. An efficient algorithm was given by [28]. A linear time algorithm was found by Ding, Filkov and Gusfield [22]. Later in this thesis, the perfect phylogeny haplotyping problem will be used to find haplotypes for the founders of a pedigree in Chapter 4.

2.2 Detecting Individuals in Pools

The problem under investigation here is the privacy of individuals participating in genetic studies. Suppose there is a cohort of n individuals contributing to a genetic study. Suppose that the genotype alleles are binary, i.e. $g_{ij}^k \in \{0,1\}$. It is possible to *pool* the genetic data of the participants as follows

$$p_j = \frac{1}{2n}\sum_{i=1}^{n} g_{ij}^0 + g_{ij}^1.$$

The pool allele frequencies p_j for site j are simply the fraction of genotypes of individuals in the pool that have the 1-allele. Originally it was thought that these pool allele frequencies, $\vec{p} = (p_1, ..., p_m)$, for a study cohort could be publicly released, in order to facilitate research, without violating the privacy of study participants.

In 2008, Homer, et al. [41] published the striking result that publicly releasing the pool allele frequencies of a cohort violated the privacy of the study participants. Indeed, they showed that, given the genotype of an individual, it is possible to detect whether that person participated in the study. Intuitively, this result extends also to close relatives, for example, siblings can share long haplotypes.

More recently, Sankararaman, et al. [81] investigated the number of independent loci whose pool allele frequencies could safely be published without violating privacy. Specifically they tested the hypothesis that an individual was a better match for a pool than for an ethnically matching population. They used a likelihood ratio test to test for the hypothesis that the individual's genotype could be generated by a random draw of two haplotypes from the pool versus the hypothesis that the individual's genotype could be generated by a random draw of two haplotypes from the ethnically matching population. They proved theoretical bounds making use of the likelihood ratio test for individuals in the pool which assumes independence of the sites in the genome. Indeed the likelihood ratio test is the theoretically most powerful test under the assumption of independence.

Here, we investigate whether the dependence between sites in the genome, or linkage disequilibrium, can be used to more reliably detect individuals in a pool than is possible under the assumption of independence. The work in the section is unpublished work done in collaboration between the author, Sankararaman and Halperin.

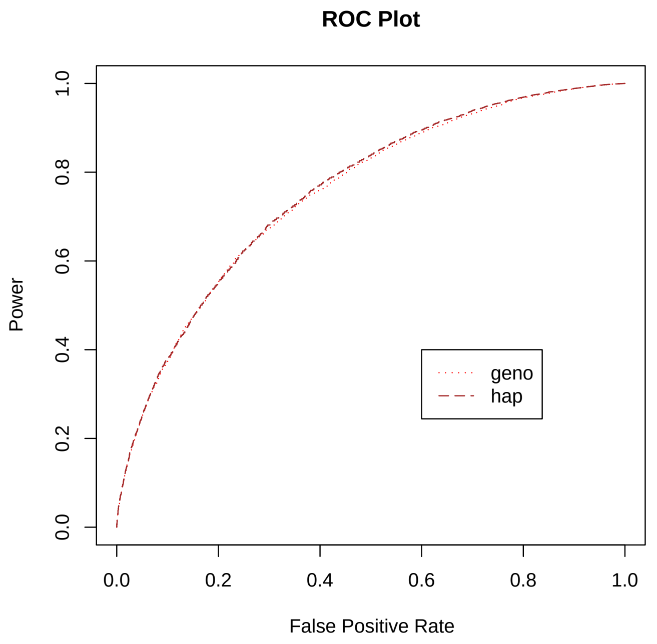


Figure 2.2: **Equivalent Power of Perfect Phylogeny Haplotypes and Independent Loci.** For common haplotypes and common alleles (0.1 frequency), both tests were applied to 1000 SNPs. The haplotype test was performed on 500 independent SNP-pairs whose haplotypes conformed to a perfect phylogeny. The genotype test had 1000 independent SNPs.

2.2.1 Likelihood-Ratio Test

Two likelihood ratio (LR) tests were computed: one for genotypes and one for haplotypes. For the genotype case, the LR test was computed exactly as done by Sankararaman, et al [81]. For SNP j, let p_j be the pool allele frequency, and let f_j be the population allele frequency. For the person of interest i, let c_{ij} be the number of 1-alleles at SNP j, meaning that $c_{ij} = g_{ij}^0 + g_{ij}^1$ for $g_{ij}^k \in \{0,1\}$. Then the genotype LR test for person i over m independent SNPs is

$$L_i^g = \sum_{j=1}^{m} c_{ij} \log\left(\frac{p_j}{f_j}\right) + (2 - c_{ij}) \log\left(\frac{1-p_j}{1-f_j}\right). \tag{2.1}$$

For k linked SNPs, the haplotype LR test was computed in a similar manner. Let $h \in \mathbb{Z}_2^k$ be a haplotype on k SNPs, i.e. a string of k bits. Divide the genome into m sets of k contiguous SNPs. For the jth set of k SNPs, let f_{jh} be the population haplotype frequency and p_{jh} be the pool haplotype frequency. Presume that we have the haplotypes of the person whose presence in the pool we wish to detect. For this person of interest i, let $c_{ijh} \in \{0,1,2\}$ be the count of haplotype h for the jth set of SNPs, where $\sum_h C_{ijh} = 2$. Then the haplotype LR test for person i at m haplotype blocks of SNPs is

$$L_i^h = \sum_{j=1}^{m} \sum_{h} c_{ijh} \log\left(\frac{p_{jh}}{f_{jh}}\right).$$

2.2.2 Perfect Phylogeny Simulation

Can linked SNPs be as useful as the same number of independent SNPs? In an ideal simulation setting, we examine the haplotype test for $k = 2$ where the number of independent SNPs for the genotype test is $m/2$ where m is the number of linked SNP-pairs in the haplotype test. We find that false positive and false negative rates for these two tests are essentially identical.

The simulation was performed independently for 1000 pairs of SNPs, and each SNP pair was in LD according to the perfect phylogeny model. Specifically, for each SNP pair, j, first select 3 haplotypes, H_{j1}, H_{j2}, H_{j3}, u.a.r. from $\{00, 01, 10, 11\}$. To randomly choose the haplotype frequencies, F_{j1}, F_{j2}, F_{j3}, draw $F_{j1} \sim Unif(0,1)$ and $F_{j2} \sim Unif(0, 1 - F_{j1})$ and set $F_{j3} = 1 - F_{j2} - F_{j3}$. These frequencies represent the population haplotype frequencies. Next, we draw 1000 haplotypes for the pool. Each person's haplotypes were drawn from the population, with independent draws at each pair of SNPs. The likelihood ratio (LR) test was computed with complete knowledge of the actual pool haplotype and allele frequencies and with known population haplotype and allele frequencies. (Also considered were the degenerate cases, where one or more $F_{jk} = 0$, however the ROC plots of the genotype and haplotype tests remained very similar.)

Consider a simulation of 10 pools with 500 perfect phylogeny tuples and 1000 independent SNPs. Let all the haplotype frequencies and all the allele frequencies be common (i.e. $0.9 \geq F_{jk} \geq 0.1$ for all j, k, and the allele frequencies, $A_{j,l} = \sum_k H_{j,k,l} F_{j,k}$, of SNP $l \in \{1, 2\}$

for tuple j, similarly satisfy $0.9 \geq A_{jl} \geq 0.1$ for all j and l). Figure 2.2 shows that the haplotype test performs similarly to the genotype test (Equation 2.1). This result means that relative to the total number of independent SNPs, half the number of haplotype blocks are necessary to obtain the same power when the haplotypes conform to the perfect phylogeny. Worded differently, using the same number of SNPs, in linked and unlinked configurations, we get the same power if the haplotypes conform to a perfect phylogeny as we do with independent SNPs.

2.2.3 Estimated Frequencies

Simulation. We simulated pools and reference populations from the 58C and UKBS control groups from the Wellcome Trust Case Control Consortium (wtccc). There were 3004 individuals genotyped on the 500k Affymetrix array, and after preprocessing the data, there were 2937 unrelated individuals with 462386 common SNPs. A SNP is said to be *not in Hardy-Weinberg equilibrium (HWE)* if the frequency of the heterozygous genotype does not match that expected from the independent allele frequencies. The preprocessing step removed SNPs that were rare or not in HWE and removed individuals that had more than 3% missing data, that had too much IBD with another individual, or that had non-European ancestry. In each replicate, a pool of n individuals were selected without replacement from the 2937 available individuals, and the $2937 - n$ unselected individuals comprised the reference population.

Estimation for $k = 2$. The haplotype frequencies for the reference population were inferred by phasing all the reference individuals together using Beagle. This procedure used all the available SNP genotypes, and was not limited to the 29624 pairs of SNPs that were selected in the preprocessing step. A test individual's haplotypes were inferred also using Beagle, by phasing that person together with the reference population.

The pool haplotype frequencies were calculated by assuming that the pool individuals were drawn from the same population as the reference individuals (and assuming correct estimation of the population haplotypes). If the least frequent population haplotype occurs < 0.005, then the perfect phylogeny method for pool haplotype inference was used. Otherwise, the pool haplotype frequencies were obtained by minimizing the Kullback-Leibler (KL) divergence between the pool haplotype frequencies and the population frequencies.

In the first case, the least frequent, or minor, haplotype was rare enough that it appears in very few pools. With the three remaining haplotypes and the pool allele frequencies, we solved a system of three equations with three unknowns to obtain the haplotype frequencies. In cases where any estimated haplotype frequency were negative, then the data were incompatible with the population perfect phylogeny model. For example, suppose the population haplotypes are $\{00, 01, 11\}$ and the pool allele frequencies are 0.3 and 0.2. Notice that if the pool haplotype frequencies are p_1, p_2, p_3, then these equations imply that $p_3 > p_2 + p_3$, which is not possible with non-negative haplotype frequencies.

When the perfect phylogeny case did not apply, either due to a common minor haplotype or due to an incompatibility, we optimized the four possible pool haplotype frequencies

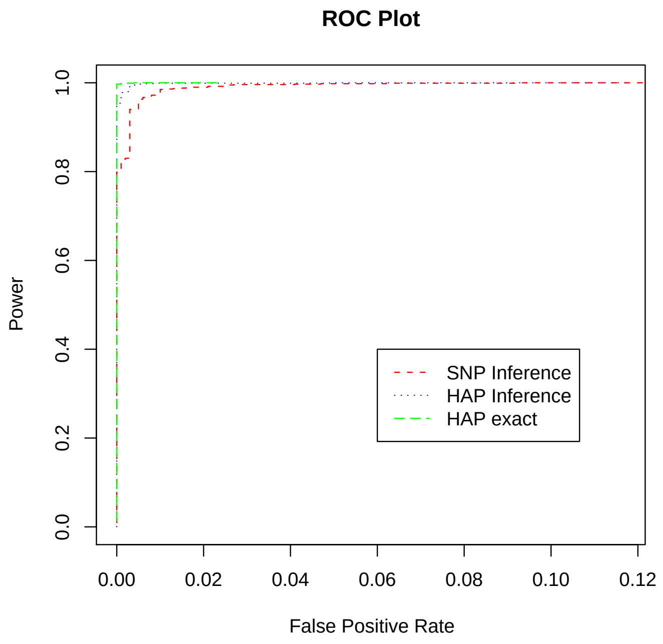


Figure 2.3: **Power of the LR Test After Haplotype Frequency Estimation with $k = 2$.** Pools of 10 individuals, each with 100 independent pairs of linked SNPs. The analysis was repeated for 100 replicates. The left-most curve is the test results when the pool haplotype frequencies are known exactly. The curve just under that one, is obtained by inferring the pool haplotype frequencies. The right-most curve is the result of the testing all 200 SNPs with the genotype test.

to minimize the KL divergence between the pool frequencies and the reference haplotype frequencies. Since we used the pool allele frequencies as constraints, there was one free variable; here refer to it as p_{01}. For tuples of linked SNPs, i.e. $k = 2$, the objective function is

$$\begin{aligned}\hat{p} &= argmin_p \sum_{p_h} p_h \log \frac{p_h}{f_h} \\ &= (1 - a_0 - p_{01}) \log \frac{1 - a_0 - p_{01}}{f_{00}} + p_{01} \log \frac{p_{01}}{f_{01}} + (a_0 - a_1 + p_{01}) \log \frac{a_0 - a_1 + p_{01}}{f_{10}} \\ &\quad + (a_1 - p_{01}) \log \frac{a_1 - p_{01}}{f_{11}}\end{aligned}$$

where, as before, $h \in \{00, 01, 10, 11\}$ is a haplotype for the SNP pair, a_0 and a_1 are the pool allele frequencies, p_h is a pool haplotype frequency, and f_h is a population haplotype frequency.

Results for 2-SNP Blocks. As before, we computed the power by counting the fraction of individuals in the pool that had a significant LR test result when $k = 1$ and $k = 2$. The

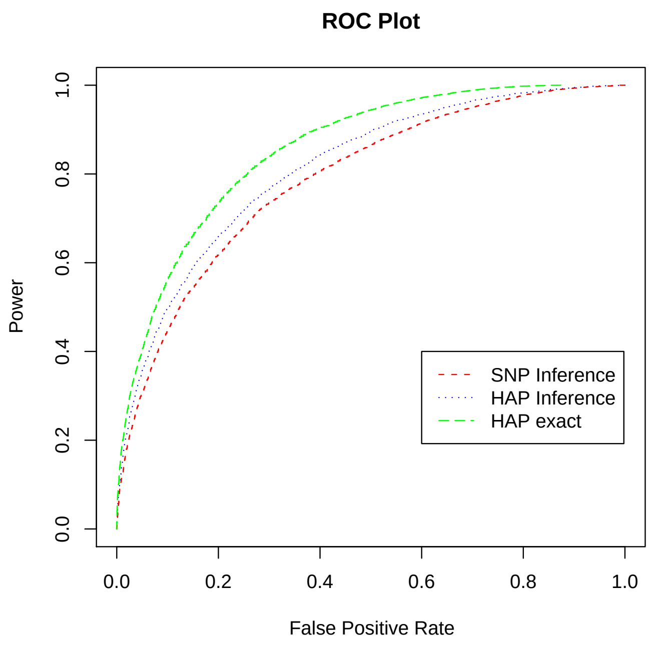


Figure 2.4: **Power of the LR Test After Haplotype Frequency Estimation for** $k = 1, 2$ This figure was generated using only the analytical minimum KL calculation (i.e. without using the perfect phylogeny test). Pools of 100 individuals, each with 100 independent pairs of linked SNPs. The analysis was repeated for 100 replicates. The left-most curve is the test results when the pool haplotype frequencies are known exactly. The curve just under that one, is obtained by inferring the pool haplotype frequencies. The right-most curve is the results of the testing all 200 SNPs with the genotype test.

false positive rate was obtained by counting the fraction of pool individuals for which there was a significant LR test result on the pool of $n-1$ individuals where the test individual's genotype had been removed from the pool.

For pools of $n = 10$ individuals, $p = 100$ pools, 100 pairs of SNPs, the genotype and haplotype inference test results are shown as the red and blue lines in Fig. 2.3. The haplotype test shown in green (third line in the legend) is the haplotype test performed when knowing the actual pool haplotype frequencies. This shows the power lost by mis-estimation of the pool haplotype frequencies.

Figure 2.4 was generated using only the minimum KL calculation (i.e. without using the perfect phylogeny test) on pools of 100 individuals, each with 100 independent pairs of linked SNPs. This plot also shows the power lost by the haplotype frequency estimates relative to knowing the actual pool haplotype frequencies.

Results for 3-SNP Blocks We used a similar inference method for the pool frequencies of 3-SNP blocks (KL equations not shown). We apply the LR test for pool membership with $k = 1, 2, 3$ and compare the results. The results shown in Figure 2.5 show that the power

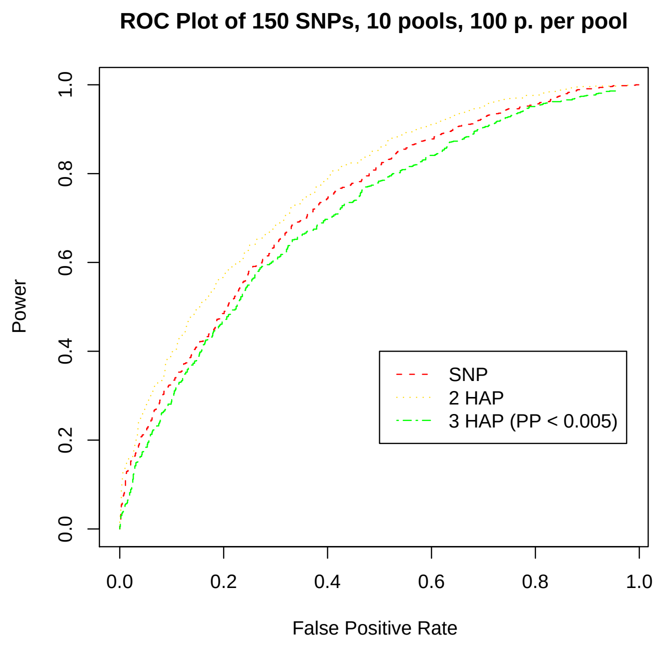


Figure 2.5: **Power of the LR Test After Haplotype Frequency Estimation for** $k = 1, 2, 3$ Comparing tests on single SNPs, tuples of SNPs, and triples of SNPs. This test was run on 150 SNPs for 10 pools with 100 individuals each.

of the 3-SNP test deteriorates below the single SNP test that assumes independence. This shows that for 3-SNP haplotypes, the potential improvement in power by taking advantage of haplotype information is lost due to errors in haplotype frequency estimation.

2.2.4 Discussion

We introduced a haplotype likelihood ratio (LR) test for detecting the presence of individuals in pools. For haplotypes on 2 SNPs, this test effectively leverages linkage disequilibrium to improve the power over that obtained by assuming SNPs are independent. For haplotypes on 3 SNPs, we find that the inaccuracy of haplotype inference weakens the power of the haplotype likelihood ratio test making it less effective than the likelihood ratio test that simply assumes that the sites are independent.

Recall that Sankararaman, et al. [81] investigated the number of independent loci whose pool allele frequencies could safely be published without violating privacy. Clearly it is critical that those published pool allele frequencies are indeed from independent sites in the genome. Otherwise a haplotype test, such as the one introduced here, might still be able to detect study participants in the pooled data.

2.3 Efficiently Constructing Perfect Phylogenies from Binary Characters with Missing Data

The work from this section was done in a collaboration between the author and Stevens [56].

For partial characters, we need to redefined the labeling map. Let the labeling map be $\phi : X \rightarrow V^2$ where V^2 are the power sets of V. A taxon can always label some connected subtree of the phylogeny, as noted by Halperin and Karp [37].

2.3.1 Background and Examples

Let $V(C_i, C_j)$ be the set of values that the pair of characters C_i and C_j takes over the the observed taxa in $\mathcal{C}$. Then for binary characters $V(C_i, C_j) \subseteq \{(0,0),(0,1),(1,0),(1,1)\}$. The classic *splits-equivalence theorem* [82] states that a collection of full binary characters is *compatible* if and only if $|V(C_i, C_j)| \leq 3$ for all pairs of characters C_i and C_j. An equivalent statement of the splits-equivalence theorem, is that each character corresponds to a single edge in the tree [82].

Definition 2.3.1 (rich data hypothesis). *A set of binary partial characters where* $|V(C_i, C_j)| = 3$ *for all pairs of characters* C_i *and* C_j *conforms to the* rich data hypothesis *(RDH) of Halperin and Karp [37].*

In general, determining if a collection of binary partial characters is compatible is NP complete. The rich data hypothesis is of theoretical importance in that it characterizes a class of tractable perfect phylogeny problems on binary partial characters. Halperin and Karp were also able to demonstrate that under certain sampling conditions data will frequently conform to the rich data hypothesis.

Halperin and Karp noted that for RDH characters there was at most one topology for the resulting phylogeny [37]. We will prove this fact later.

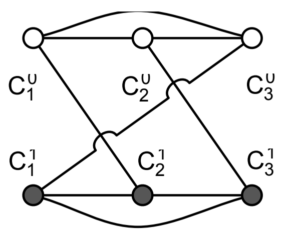
Partition and Subtree Intersection Graphs

There are two related notions that are extremely important for determining character compatibility. As defined by Semple and Steel, we introduce the *partition intersection graph* $int(\mathcal{C})$ and the *subtree intersection graph of* $\mathcal{T}$ *induced by* $\mathcal{C}$ $int(\mathcal{C}, T)$.

Each character in $\mathcal{C}$, $C : X' \rightarrow A$, induces a partition on the taxa given by $C^{-1}(A)$. The *partition intersection graph* $int(\mathcal{C}) = \{V_I, E_I\}$ summarizes the co-occurrence of character state pairs on the observed taxa. $int(\mathcal{C})$ is defined on $\mathcal{C}$ as follows. There is a vertex in V_I for each character, state pair. For a binary character C_i there will be two vertices corresponding to C_i^0 and C_i^1. There is an edge between two vertices if their character state pairs occur for some taxon s in X. In other words $(C_i^a, C_j^b) \in E_I$ if there is some taxon s such that $C_i(s) = a$ and $C_j(s) = b$.

The partitions induced by character C can be mapped to the phylogenetic tree $\mathcal{T}$ using ϕ. We can define an intersection graph on the vertex set of the minimal subtree of $\mathcal{T}$ containing a partition of character C. Specifically, the *subtree intersection graph of* $\mathcal{T}$ *induced by* $\mathcal{C}$

	C_1	C_2	C_3
S_1	0	0	0
S_2	?	0	0
S_3	1	?	0
S_4	?	0	1
S_5	1	1	1



	C_1	C_2	C_3
S_1	0	0	?
S_2	?	0	0
S_3	0	1	0
S_4	0	?	1
S_5	1	?	0
S_6	1	1	1

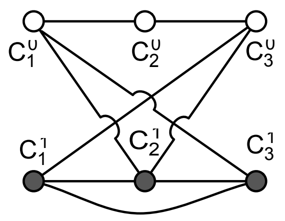


Figure 2.6: Two examples of a set of characters $\mathcal{C}$ that meets the RDH requirement but does not have a perfect phylogeny. To the left of each is the partition intersection graph $int(\mathcal{C})$ which contains cycles of four vertices that violate Buneman's theorem.

$int(\mathcal{C}, T) = \{V_I, E_J\}$ is the graph having vertex set V_I and edges E_J. As defined above, there is a node in V_I for every character, state pair. Let A be a set of taxa, and let $T(\phi(A))$ be the minimal subtree of T containing vertex set $\phi(A)$. Then there is an edge $(C_i^a, C_j^b) \in E_J$ if there is some tree-node $v \in T$ such that $v \in T(\phi(C_i^a))$ and $v \in T(\phi(C_j^b))$

A *legal edge* in either of these graphs must connect nodes belonging to different characters.

Some RDH examples and intuition

Not every collection of binary partial characters satisfying the rich data hypothesis is compatible. While the partial characters do not violate the splits equivalence, all resolutions will violate splits equivalence. Figure 2.6 gives two examples of binary matrices that satisfy the rich data hypothesis but no completion of these characters is compatible.

Not every set of compatible characters satisfies the rich data hypothesis. For example, let the character set $\mathcal{C} = \{C_1, C_2, C_3, C_4, C_5\}$. The taxa are $X = \{S_1, S_2, S_3\}$. The tree and character matrix are given in Figure 2.1 and Table 2.1 (example taken from [74]). Indeed in this example, it is easy to see that the character matrix does not satisfy the RDH (see C_1 and C_2), even though there exists a tree on which all the characters are convex. Recall the splits-equivalence theorem, which states that there is a correspondence between the edges of $\mathcal{T}$ and the characters. This theorem states that for each character, C there is an edge which when removed from the tree produces two subtrees, one subtree containing only the taxa mapped to state 0 by C and the other subtree containing only taxa mapped to state 1 by C.

This example illustrates several aspects with the RDH with distinct informative characters. If a character appears twice in the character matrix, then the matrix fails the rich

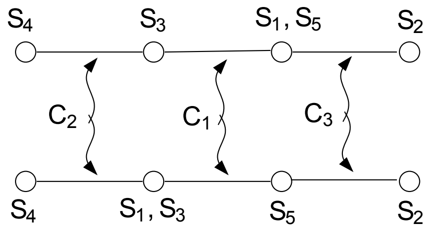


Figure 2.7: Two trees that fit the partial characters given in Table 2.2. The edges are labeled with the characters they correspond to.

data hypothesis test. In addition, if every taxon has the same state in character C_1, then the matrix will fail the RDH test.

Lemma 2.3.1. *Let $\mathcal{C}$ be a set of partial characters on taxa X such that $\mathcal{C}$ satisfies the RDH. There exists a set $\mathcal{C}$ for which there are multiple resolutions for some unspecified character state.*

Proof. Let $\mathcal{C} = \{C_1, C_2, C_3\}$ be the set of partial characters and let the taxa be $X = \{S_1, S_2, S_3, S_4, S_5\}$ as given in Table 2.2. $\mathcal{C}$ satisfies the RDH since all entries S_k from a pair of columns $C_i(S_k) \times C_j(S_k) \in \{\{0,0\},\{0,1\},\{1,0\}\}$

There is no unique resolution for the unspecified character state $C_1(S_1)$. There are two trees that satisfy the compatibility requirement. These trees are given in Figure 2.7 with each edge being labeled by the character it corresponds to. □

	C_1	C_2	C_3
S_1	*	0	0
S_2	1	0	1
S_3	0	0	0
S_4	0	1	0
S_5	1	0	0

Table 2.2: $\mathcal{C}$ has no unique resolution for unspecified state $C_1(S_1)$

2.3.2 Enumerating Resolutions for Binary Partial Characters under the RDH

When given data that satisfies the *rich data hypothesis*, we can find a tree that is compatible with the partial characters in $O(nm^2)$ time. Notice that the RDH requires that the input characters all be distinct.

Algorithm 2 defines the BINARY-RESOLVE algorithm. The algorithm takes as input a set of m distinct partial characters on n taxa such that every pair of characters satisfies the rich data hypothesis. This algorithm is a modified version of the linear-time *tree-popping* algorithm first described by Meacham [69] and reviewed in full detail in *Phylogenetics* [82].

The BINARY-RESOLVE algorithm works by iteratively modifying a tree to respect the bipartition on the taxa that is described by character C_i. The tree begins as a single node labeled with all the taxa. For each new character, we find the unique node, b, that is labeled with taxa having both character states $0, 1$ of C_i. We then add a single edge to the tree by splitting b into two nodes b_0 and b_1, each labeled with the taxa of b that belong to a single color or bipartition of C_i. The edges incident to b, are then connected to either b_0 and b_1 in a manner preserving convexity. Consider $T[V\backslash\{b\}]$, the subtrees induced by removing node b from T. Each of these subtrees that is 0-colored is connected to b_0 and similarly for b_1. This splitting operation is called *tree-popping.*

Compatibility under the RDH

Before describing the details of the BINARY-RESOLVE algorithm, we need to define several terms and present several results regarding compatibility under the RDH.

Recall that ϕ is defined as mapping a taxa to a set of nodes in the tree where the set must define a subtree of T. For convenience, we define $\phi^{-1}(T_v|_{(u,v)}) = \{s : \phi(s) \subseteq T_v|_{(u,v)}\}$.

A character C_i defines a 2-coloring on the subtrees of the phylogeny $\mathcal{T} = (T = (V, E), \phi)$ as follows. Let (u, v) be an edge in the tree T. Let $T_v|_{(u,v)}$ be the subtree of T, rooted at v and not including edge (u, v). Given an edge $e = (u, v) \in E$ and a *coloring character*, C_i, there is a *coloring of the subtrees*, T_u and T_v, obtained by removing edge (u, v) from the graph. The *coloring* is given by the resolved character states of C_i induced on T_u and T_v. In other words, for a directed edge $(u, v) \in E$, we define

$$\mathit{color}(T_v|_{(u,v)}, C_i) = \{C_i(s) | C_i(s) \neq *, \phi(s) \subseteq T_v|_{(u,v)}, \text{ and } s \in X\}.$$

If a subtree is labeled with a single character state, we call it *monochromatic*, otherwise the subtree is *bicolored.* Subtrees having only unspecified character states are said to have *no color.*

Similarly, we can define the *color of a node v* with respect to coloring character C_i to be

$$\mathit{color}(v, C_i) = \bigcap_{u \in adj(v)} \mathit{color}(T_v|_{(u,v)}, C_i),$$

where $adj(v)$ are the nodes adjacent to u in tree T and $T_v|_{(u,v)}$ is the subtree of T rooted at v obtained to removing edge (u, v) from the graph. Again, we can discuss a node that is monochromatic, bicolored, or has no color.

Now, we will prove the essential property of a set of compatible characters that satisfy the RDH. The first result, that there are no uncolored nodes, is critical for the intuition behind the tree-popping algorithm.

Theorem 2.3.1. *Given that characters $C_1, C_2, ..., C_i$ satisfy the RDH with a character C_i to server as a coloring character and phylogenetic tree $\mathcal{T} = (T = (V, E), \phi)$ that is compatible with distinct characters $\{C_1, C_2, ..., C_{i-1}\}$, every vertex in the tree has a color (i.e. it is* *impossible to select a directed edge (u, v) such that subtree T_v rooted at v contains only unresolved characters, and it is impossible for a vertex, v, to have $color(v, C_i) = \emptyset$).*

Proof. First, we prove the statement about subtrees being resolved. Assume that there exists a directed edge $(u, v) \in E$ such that $color(T_v, C_i) = *$. Then the set of taxa $S_v = \phi^{-1}(T_v|(u, v)) = \{s : \phi(s) \subseteq T_v|_{(u,v)}\}$ mapped to the subtree T_v have no color, $C_i(s) = *$ for all $s \in S_v$. Then, by the splits-equivalence theorem, there is a resolved character C^R_{uv} that corresponds exactly to the edge (u, v) that splits the taxa into the partitions $\phi^{-1}(T_u|_{(v,u)})$ and $\phi^{-1}(T_v|_{(u,v)})$; see Figure 2.8. But, since all of the partition C_i that overlaps with T_v is unspecified, the character pair C_i and C^R_{uv} fail the rich data hypothesis. This means that for any unresolved character $C_{u,v}$, the pair C_i and C_{uv} also fail the rich data hypothesis. This contradiction proves that all subtrees must have a color.

Second, we prove that $\forall v \in V$, $color(v, C_i) \neq \emptyset$ by contradiction. Assume there exists a vertex v for which $color(v, C_i) = \emptyset$. Then there must exist two subtrees $T_v|_{(u,v)}$ and $T_v|_{(w,v)}$ rooted at v such that

$$color(T_v|_{(u,v)}, C_i) \cap color(T_v|_{(w,v)}, C_i) = \emptyset.$$

Since we know that $color(T_v|_{(u,v)}, C_i) \neq \{*\}$ and $color(T_v|_{(w,v)}, C_i) \neq \{*\}$, the first part of the proof, then it must be the case that one of these subtrees has taxa labeled with 0 and the other taxa labeled with 1. With out loss of generality, let

$$\{1\} \subset color(T_v|_{(u,v)}, C_i), \text{ and } \{0\} \subset color(T_v|_{(w,v)}, C_i).$$

Then there can be no taxa labeling node v ($\forall s, \phi(s) \neq \{v\}$). Therefore, for any full resolution of the characters $\mathcal{C}^R$, by splits equivalence, there must be (at least) two edges with the same split, i.e. the edges (u, v) and (w, v). Therefore, two full characters must be identical, which violates the RDH for those two characters. This also means that any partial characters of those characters must also violate the RDH. Therefore, every node must be colored. $\square$

Theorem 2.3.2. *A phylogenetic tree $\mathcal{T} = (T = (V, E), \phi)$ is compatible with partial characters $\mathcal{C} = \{C_1, C_2, ..., C_m\}$ if and only if every full resolution, $\mathcal{C}^R$, is compatible with $\mathcal{T}$ where for $C_i \in \mathcal{C}$, $C_i^{\mathcal{R}} \in \mathcal{C}^R$ is defined by $C_i^{\mathcal{R}}(s) = color(v, C_i)$ for some $v \in \phi(s)$ with coloring character C_i.*

Proof. Given a phylogenetic tree $\mathcal{T} = (T = (V, E), \phi)$ compatible with partial characters $\mathcal{C} = \{C_1, C_2, ..., C_m\}$, we will prove that $\mathcal{C}^R$ is compatible with $\mathcal{T}$. First, we will show that the above $C_i^{\mathcal{R}}$ is a well defined resolution of C_i, then we will show that it is compatible with $\mathcal{T}$.

First, for $C_i^{\mathcal{R}}$ to be a resolution of C_i, it must be that $\forall s$ such that $C_i(s) \neq *$ we have $C_i(s) = C_i^{\mathcal{R}}(s)$. Certainly, we know that $\forall s$ such that $C_i(s) \neq *$, $C_i(s) \in color(T_v|_{(u,v)}, C_i)$ for all v such that $\phi(s) \subseteq T_v|_{(u,v)}$ by definition of the coloring. Since character $C_i(s)$ must be convex on the tree and since $color(T_v|_{(u,v)}, C_i) \neq *$ by the previous lemma, this implies that $C_i(s) \in color(v, C_i)$ for all $v \in \phi(s)$.

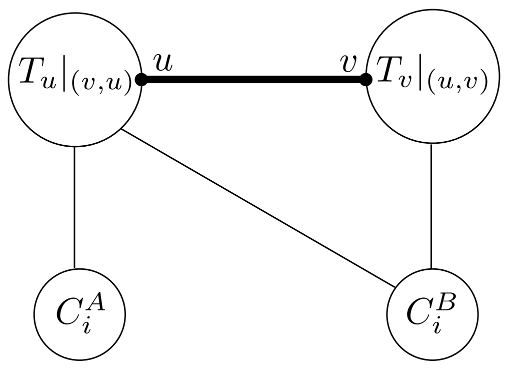


Figure 2.8: The heavy edge, (u, v), is a tree edge. The light edges are the pair-wise partition intersection graph edges. C_i^A represents the A-partition of C_i, $\{s|C_i(s) = A\}$, where $A \in \{0, 1\}$, similarly for C_i^B. Notice that coloring character C_i results in the subtree $T_u|_{(v,u)}$ being bi-colored while subtree $T_v|_{(u,v)}$ is monochromatic.

Now, to show that $C_i^{\mathcal{R}}$ is well-defined, we must show that every taxon s has a single color. By the previous lemma, every taxon has some color (i.e. there are no uncolored taxa, since there are no uncolored nodes). Therefore, we need only establish that every taxon, s, has no more than one color, i.e. that $|\{a \in \mathit{color}(v, C_i)|v \in \phi(s), a \neq *\}| = 1$. This is equivalent to showing that $\forall v \in V$, $|\{a \in \mathit{color}(v, C_i)|a \neq *\}| = 1$, meaning that there are no bi-colored nodes. We will show this by contradiction. Let v be a bi-colored node, i.e. a node v such that $\{0, 1\} \in \mathit{color}(v, C_i)$. This implies that $\{0, 1\} \in \bigcap_{u \in adj(v)} \mathit{color}(T_v|_{(u,v)}, C_i)$. In turn, this means that $\{0, 1\} \in \mathit{color}(T_v|_{(u,v)}, C_i)$ $\forall u \in adj(v)$. Therefore C_i is not convex on $\mathcal{T}$ for any possible resolution, since clearly any possible character-state subtrees T_0 and T_1 are not disjoint.

Now, given full characters $\mathcal{C}^R$ compatible with $\mathcal{T}$, then any partial character set $\mathcal{C}$ that satisfies the RDH is also compatible with $\mathcal{T}$ by definition. □

This theorem tells us three things. First, when given a set of characters, specifying T and ϕ is the same as specifying compatible resolutions of the characters. Recall that there may be multiple resolutions. Second, for s with $\phi(s)$ containing multiple nodes, s can take the color of any of those nodes. Third, we can define a correspondence between characters and edges by appeal to the splits-equivalence theorem on full characters. We will formalize this next.

For a particular tree T, using Theorem 2.3.2 we can define a one-to-one mapping f_T from the partial characters $\mathcal{C}$ to the resolved characters $\mathcal{C}^R$. This is well-defined, because for each character $C_i \in \mathcal{C}$, simply apply the theorem to color the taxa, thereby obtaining a resolved character. Since $\mathcal{C}^R$ is compatible with $\mathcal{T}$, the splits-equivalence theorem guarantees us a correspondence between edges in the tree and resolved characters. Let $s : \mathcal{C}^R \to E$ be the mapping from edges of $\mathcal{T}$ to characters given by splits-equivalence.

Definition 2.3.2 (correspondence of partial characters to edges)**.** *For phylogenetic tree $\mathcal{T} = (T = (V, E), \phi)$, let $g : \mathcal{C} \to E$ be a function giving the correspondence between edges* *in the tree and partial characters. Specifically, we can write g as the composition of two functions: $g = s \circ f_T$. We call $g(C_i)$* the edge that corresponds to partial character $C_i \in \mathcal{C}$. *A set of partial characters may be mapped to a particular edge $g^{-1}(e)$ for $e \in E$.*

Before proceeding to the main result in this section, we need to extend our notion of a partition intersection graph and prove one more lemma. We extend the idea of a partition intersection graph to consider taxa that may be unresolved for a particular character.

We define a *pair-wise partition intersection graph (PWPIG) under $\mathcal{T}$* as being a partition intersection graph on two characters: $C_{(u,v)}$ represented by an edge (u, v) in the tree and some other character C_i. Consistent with the definition of a partition intersection graph for a set of characters, this pair-wise graph is a bipartite graph with vertex set $W = X \cup Y$ where $X = \{T_u|_{(v,u)}, T_v|_{(u,v)}\}$, and $Y = \{C_i^0, C_i^1\}$. Again, we have edges (x, C_i^A) where $x \in X$ and $C_i^A \in Y$ if and only if there is a taxon $s \in \phi^{-1}(x)$ and $C_i(s) = A$, for $A \in \{0, 1\}$. This object is a hybrid of the partition intersection graph and the subtree intersection graph.

Corollary 2.3.1. *Given a character C_i to serve as a coloring character and a phylogenetic tree $\mathcal{T} = (T = (V, E), \phi)$ compatible with characters $\{C_1, C_2, ..., C_{i-1}\}$, then there is at most one bicolored node b.*

Proof. There is at most one bicolored node, b, by contradiction. Let b' be another such node. Pick an arbitrary edge e' on the undirected path between $b \rightsquigarrow b'$. Let $C_{e'} \in g^{-1}(e')$ be one corresponding character for edge e', by splits-equivalence on the resolved characters. Then in the pair-wise partition intersection graph for $C_{e'}$ and C_i, there are two bicolored subtrees. Therefore there are four edges in the graph, and the characters are incompatible. Therefore, we conclude that there is at most one bicolored node. □ □

Theorem 2.3.3. *Given a character C_i to serve as a coloring character and a phylogenetic tree $\mathcal{T} = (T = (V, E), \phi)$ compatible with characters $\{C_1, C_2, ..., C_{i-1}\}$ where for all $s \in X$ the subtrees $\phi(s)$ are of maximal compatible size, then there is a unique bicolored-node $b \in V$ if and only if*

1. *C_i is not already represented by an edge in the tree T, and*
2. *C_i is compatible with $\{C_1, C_2, ..., C_{i-1}\}$ on tree $\mathcal{T}' = (T', \phi')$ with ϕ' having subtrees of maximal compatible size where $T' = (V \setminus \{b\} \cup \{b_0, b_1\}, E \cup \{(b_0, b_1)\})$ and*

$$\phi'(s) = \begin{cases} \phi(s) & \forall s \text{ s.t. } b \neq \phi(s) \\ \phi(s) \setminus \{b\} \cup \{b_0, b_1\} & \forall s \text{ s.t. } b \in \phi(s) = b \text{ and } C_i(s) = * \\ \phi(s) \setminus \{b\} \cup \{b_{C_i(s)}\} \setminus \{\tau_1, ..., \tau_k\} & \forall s \text{ s.t. } b \in \phi(s) = b \text{ and } C_i(s) \neq * \end{cases}$$

where for each i with $1 \leq i \leq k$, $\tau_i \in T[V \setminus \{b\}]$ such that color$(\tau_i, C_i) \neq C_i(s)$, and $T[V \setminus \{b\}]$ is subtrees of T induced by removing node b.

Proof. For each C_j such that $j < i$, there is an edge $e = g(C_j)$ in the tree T that corresponds to character C_j, by splits-equivalence on the resolved characters. This edge represents the partitions of character C_j, since the taxa on the two subgraphs induced by removing e

correspond exactly to the partitions the character C_j induces on the taxa. This tree edge is drawn in Figure 2.8.

($\Leftarrow$) First, we assume that characters $\{C_1, C_2, ..., C_i\}$ are compatible and C_i not in the tree. Then for all $j < i$, we have characters C_j and C_i whose partition intersection graph contains exactly 3 edges (by the RDH and by compatibility). Notice that there is a full resolution of the characters $\{C_1, ..., C_{i-1}\}$ specified by $\mathcal{T}$. Let edge $e = g(C_j)$ correspond to character C_j in tree T by splits-equivalence for the resolved characters. Directed edge e towards the bi-colored subtree of T induced by the removal of edge e and according to coloring character C_i. By the RDH, there is one such bi-colored subtree, so the direction is well defined. Furthermore, all edges e have a direction, since they all have corresponding characters (by the previous theorem). Then there must exist at least one node with out-degree zero, since the RDH implies that all the characters are distinct and since character C_i is not in tree T. Every node with out-degree zero is clearly bicolored, since $\{0, 1\} \subset \mathit{color}(T_v|_{(u,v)}, C_i)$ $\forall u \in \mathit{adj}(v)$. There is at most one node, b, with out-degree zero, by Lemma 2.3.1 since there is at most one bicolored node.

($\Rightarrow$) Second, we will assume that the node $b \in V$ is the unique bi-colored node. The challenge is to find a phylogeny $\mathcal{T}' = (T', \phi')$ that is a modification of the phylogeny $\mathcal{T}$ and on which all the characters are compatible. The subtrees of T induced by removing b, $T[V \backslash \{b\}]$, are monochromatic. We can see that any unresolved taxa s of C_i (i.e. $C_i(s) = *$), if it is mapped to one of the monochromatic induced subtrees, must be resolved to be consistent with that subtree's color by Theorem 2.3.2.

We obtain $\mathcal{T}'$ as specified in the Theorem statement. To complete the proof, we need to prove maximality of the subrees of ϕ' and compatibility of $\mathcal{T}'$.

Now, we prove maximality. For each $s \in X$, if the subtree of $\phi(s)$ is of maximal size. Since the creation of ϕ' only removes nodes from subtrees τ_i that are incompatible with $C_i(s)$, then the subtree of $\phi'(s)$ is of maximal size.

To prove compatibility, we use the pair-wise partition intersection graphs under $\mathcal{T}$ and $\mathcal{T}'$. Consider first the PWPIG under $\mathcal{T}$. For an arbitrary edge e in T, we can consider the partition intersection graph of a character $C_e \in g^{-1}(e)$, a corresponding character to e, and C_i. As above, $T_u|_{(v,u)}$ and $T_v|_{(u,v)}$ are the two subtrees of T induced by removing edge e. Without loss of generality, let subtree $T_u|_{(v,u)}$ be the subtree for which the partition intersection graph has 2 edges, one to each set of C_i. Then, we can draw the partition intersection graph as in Figure 2.8.

The PWPIG under $\mathcal{T}'$ has the same edges as the PWPIG under $\mathcal{T}$. Clearly edges in the PWPIG under $\mathcal{T}$ must appear in the PWPIG under $\mathcal{T}'$ because $\mathcal{T}'$ is a modification of $\mathcal{T}$. We need only establish that no additional edges appear; this is argued by contradiction. Suppose that under $\mathcal{T}'$, the PWPIG has an edge between C_i^A and $\phi'(T_v|_{(u,v)})$, i.e. the missing edge in Fig. 2.8. Then there exists an s such that $\phi'(s) \subseteq T_v|_{(u,v)}$ with $C_i(s) = A$. Since $b \notin T_v|_{(u,v)}$, $\phi'(s) = \phi(s)$ and $\phi(s) \subseteq T_v|_{(u,v)}$. Then under $\mathcal{T}$, the PWPIG must also have an edge between C_i^A and $T_v|_{(u,v)}$. But this edge does not exist by the RDH, and we have a contradiction. □ □

The above theorem essentially gives the tree-popping algorithm. By initially having a tree with a single node and ϕ mapping all the taxa to that one node, we can take advantage of the recursive nature of the theorem. Each step of the algorithm adds another character to the tree by splitting the sole bicolored node and updates each $\phi(s)$ to maintain the maximal subtree on which taxon s is compatible. The only remaining result is the proof of splits-equivalence for the partial characters, i.e. the proof that the tree resulting from the tree-popping method is unique.

Theorem 2.3.4. *Let $\mathcal{C}$ be a set of partial characters on taxa X such that $\mathcal{C}$ satisfies the rich data hypothesis and $\mathcal{C}$ has a perfect phylogeny. While there may be multiple resolutions for unspecified character states, the tree topology $T = (V, E)$ on which the characters are convex is unique.*

Proof. According to the tree-popping procedure, we initially have a tree with a single node and ϕ mapping all the taxa to that one node. This initial tree is unique. For each tree-popping step, a unique node b is found and split, producing a unique tree. Since the tree-popped tree is unique at every step, the final tree is unique. □ □

Algorithms

The algorithms implement the tree-popping procedure outlined in Theorem 2.3.3. When given the existing tree T and a character C_i, Algorithm 1: SPLIT-NODE finds the unique bicolored node b if it exists. Algorithm 2: BINARY-RESOLVE implements the tree-popping algorithm.

Algorithm 2: BINARY-RESOLVE implements the tree-popping algorithm. The input is the binary partial characters, $\mathcal{C} = \{C_1, C_2, ..., C_m\}$ on taxa $X = \{S_1, S_2, ..., S_n\}$ and the output is a phylogenetic tree $\mathcal{T} = (T = (V, E), \phi)$ or $\emptyset$ The initial phylogeny is simply one node labeled with all the taxa. The algorithm considers each character, C_i, in turn, adding it to the the existing phylogenetic tree. It calls SPLIT-NODE to determine which node of the tree should be tree-popped and replaced with two new nodes, one for each state of character C_i.

The input to Algorithm 1: SPLIT-NODE is a phylogenetic tree $\mathcal{T} = (T = (V, E), \phi)$ on characters $C_1, ..., C_{i-1}$, the current coloring character, C_i, and the *color* data structure containing the colors of monochromatic subtrees for each character in $\mathcal{T}$, i.e. $C_1, ..., C_{i-1}$. The output is (b, e) such that $b \in V \cup \emptyset$ is a vertex b with out-degree zero, $e \in E \cup \emptyset$ is an edge $e = (u, v)$ with no direction, the edge directions for every edge $e \in E$, $dir(e)$, and the color data structure, $color(v, C_i)$, with the color of monochromatic subtrees rooted at v under C_i. The algorithm finds the unique bicolored node by using the coloring character C_i to give directions to the edges in the tree and then finding the unique node with out-degree zero. In order to properly implement the ϕ update procedure in the Theorem, SPLIT-NODE also records the color of every monochromatic subtree in the phylogeny. All this is done in $O(mn)$ time.

Lemma 2.3.2. *For each character C_i and tree $\mathcal{T}$ the* SPLIT-NODE *algorithm finds a unique bicolored node b to tree-pop, determines that the character C_i is already in the tree, or declares the characters incompatible.*

Algorithm 1 SPLIT-NODE($\mathcal{T} = (T = (V, E), \phi), C_i, color$)

```
1: foreach edges $(u, v) \in E$ do
2:   Let $C_{uv} \in \{C_1, ..., C_{i-1}\}$ be the character corresponding to edge $(u, v)$.
3:   { Examine the PWPIG for $C_{uv}$ and $C_i$ to obtain the edge direction. }
4:   $D \leftarrow color(u, C_{uv})$
5:   Initialize $bcounts[i] \leftarrow 0$ for $i \in \{0, 1\}$.
6:   Initialize $counts[i][j] \leftarrow 0$ for $i, j \in \{0, 1\}$.
7:   foreach taxon $s \in X$ do
8:     if $count[C_{uv}(s)][C_i(s)] == 0$ then
9:       $bcount[C_{uv}] + +$
10:    end if
11:    $count[C_{uv}(s)][C_i(s)] + +$
12:  end for
13:  if $bcount[D] == 1$ then
14:    $dir((u, v)) = (u \rightarrow v)$
15:    $color(u, C_i) = j$ such that $count[D][j] > 0$
16:  end if
17:  if $bcount[1 - D] == 1$ then
18:    $dir((u, v)) = (v \rightarrow u)$
19:    $color(v, C_i) = j$ such that $count[1 - D][j] > 0$
20:  end if
21:  if $dir((u, v)) = \emptyset$ then
22:    $e = (u, v)$
23:  end if
24: end for
25: $nb \leftarrow 0$
26: foreach vertex $v \in V$ do
27:   if $color(v, C_i) = \emptyset$ then
28:     $b = v$
29:     $nb + +$
30:   end if
31: end for
32: if $nb == 0$ then
33:   { Character $C_i$ is already in the tree. }
34:   RETURN $(\emptyset, e, dir, color(., C_i)$
35: else
36:   if $nb == 1$ then
37:     RETURN $(b, \emptyset, dir, color(., C_i))$
38:   else
39:     { Character $C_i$ is incompatible. }
40:     RETURN $(\emptyset, \emptyset, dir, color(., C_i))$
41:   end if
42: end if
```

Algorithm 2 BINARY-RESOLVE($\mathcal{C}$) Returns one tree topology T and a mapping ϕ that has multiple resolutions for the taxa.

```
1: { Initialize $V$, $E$, $\phi$ with a single node. }
2: $V \leftarrow \{v\}$
3: $E \leftarrow \emptyset$
4: for $i = 1$ to $n$ do
5:    $\phi(S_i) \leftarrow v$
6: end for
7: $\mathcal{T} \leftarrow (T \leftarrow (V, E), \phi)$
8: $color(v, C_i) \leftarrow \emptyset \ \ \forall v \in V, \forall i \in \{1, ..., m\}$
9: { Process characters. }
10: for $i = 1$ to $n$ do
11:    $(b, e, dir, color(., C_i)) \leftarrow$ SPLIT-NODE($\mathcal{T} = (T = (V, E), \phi, C_i, color)$)
12:    if $b = \emptyset$ and $e = \emptyset$ then
13:       RETURN $\emptyset$ { incompatible }
14:    end if
15:    if $b \neq \emptyset$ then
16:       { Tree-pop $b$ to split it into two nodes }
17:       $E \leftarrow E \cup \{(b_0, b_1)\}$
18:       $V \leftarrow V \backslash \{b\} \cup \{b_0, b_1\}$
19:       $color(b_0, C_i) = 0$
20:       $color(b_1, C_i) = 1$
21:       foreach $u$ such that $(u, b) \in E$ do
22:          Replace edge $(u, b)$ with $(u, b_{color(u,C_i)})$
23:       end for
24:    end if
25:    { Update taxon labels to be on the correct side of the new split }
26:    foreach taxa $S_j \in X$ such that $b \in \phi(S_j)$ do
27:       if $C_i(S_j) = *$ then
28:          $\phi(S_j) = \phi(S_j) \setminus \{b\} \cup \{b_0, b_1\}$
29:       else
30:          $\phi(S_j) = \phi(S_j) \setminus \{b\} \cup \{b_{C_i(S_j)}\}$
31:          foreach node $v \in \phi(S_j)$ do
32:             if $color(v, C_i) \neq C_i(S_j)$ then
33:                $\phi(S_j) = \phi(S_j) \setminus \{v\}$.
34:             end if
35:          end for
36:       end if
37:    end for
38: end for
39: RETURN $\mathcal{T} = (T, \phi)$
```

Proof. If the algorithm finds no bicolored node, then the edge e returned by the SPLIT-NODE algorithm is the edge in the tree that corresponds to character C_i, and the tree need not be changed. Lines 25-31 of SPLIT-NODE count the number of nodes with out-degree zero, i.e. the number of nodes having no monochromatic color under coloring character C_i. If the algorithm finds a unique node b with out-degree zero, which by the proof of Theorem 2.3.3 must be the unique bicolored node, then SPLIT-NODE returns b. If SPLIT-NODE finds more than one node with out-degree zero, By Cor. 2.3.1, it declares the characters incompatible. □

Theorem 2.3.5. *If the characters are compatible, the* BINARY-RESOLVE *algorithm finds a unique tree T and for each taxon $s \in X$ the maximal subtree $\phi(s)$ of T on which the taxon is compatible.*

Proof. If the characters are compatible, by Lemma 2.3.2, for each character C_i the algorithm will find a unique bicolored node to tree-pop. The BINARY-RESOLVE algorithm implements the tree-popping procedure given in Theorem 2.3.3 for replacing node b with b_0 and b_1, and updating ϕ. Therefore by Theorem 2.3.3 the algorithm obtains the tree T which is compatible with all the characters and the ϕ which has the maximal subtrees for each of the taxa. By splits-equivalence, the tree obtained by the algorithm has the unique topology which is compatible with the input characters. □

This algorithm gives us a method to determine a unique tree that is compatible with the given partial characters, and a way to determine all the possible resolutions of the unspecified character state.

The running time of the BINARY-RESOLVE is $O(nm^2)$ time. The **for** loop at line 9 ranges over all m characters and performs two operations: SPLIT-NODE and the tree-popping procedure. The SPLIT-NODE operation which takes $O(nm)$ time. The tree-popping update for ϕ takes $O(nm)$ time. Therefore the running time of BINARY-RESOLVE is $O(nm^2)$.

Discussion

This section presents an algorithm that can calculate a unique tree that is compatible with all the binary partial characters given to it. It also finds all possible labellings for the unspecified character states. One interesting direction for future work is the extension of the rich data hypothesis to non-two-state characters. In this case, the rich data hypothesis could be defined in terms of the partition intersection graph and cycles. A pair of characters would satisfy the *generalized rich data hypothesis* if and only if adding a single edge to the partition intersection graph would create a cycle. Then all the partial characters in the data would be pair-wise compatible. This does not guarantee the compatibility of the set of characters, but it ensures that there are a restricted number of resolutions for the unspecified character states.

The algorithm presented in this section is likely to be useful for resolution of partial haplotypes. Halperin and Karp showed that the rich data hypothesis holds with high probability for a large number of partial haplotypes [37]. To get this result they created a probabilistic model that generates partial haplotypes. In addition, they showed that the phase of genotypes can be resolved by resolving the partial haplotypes.

Furthermore this method will be useful for computing probabilities of data under the coalescent with infinite sites when the data has missing values. To do this, one would need to use the enumeration algorithm here to enumerate the perfect phylogenies, and then for each perfect phylogeny compute the probability of the data under the coalescent. An efficient method for the last step is given in [101]. This enumeration algorithm is the first known algorithm allowing computation of these probabilities.

Chapter 3

Related Individuals

Pedigrees, or family trees, are important in genetics, computer science and statistics. The pedigree graph encodes all the possible Mendelian inheritance options, and provides a model for computing inheritance probabilities for haplotype or genotype data. Recent contributions to genetics from pedigree calculations include fine-scale recombination maps for humans [20], discovery of regions linked to Schizophrenia [70], discovery of regions linked to rare Mendelian diseases [71], and insights into the relationship between cystic fibrosis and fertility [31].

Algorithms for pedigree problems are of great interest to the computer science community, in part because of connections to machine learning algorithms, optimization methods, and combinatorics [32, 102, 76, 62, 91]. The pedigree graph encodes a set of inheritance possibilities and is a graphical model which models inheritance probabilities by using a graph whose edges are conditional probability events and whose nodes are random variables. Even thirty years after the development of some of the first pedigree algorithms [59, 27], pedigree graphical models continue to be a challenging graphical model to work with. Known algorithms for inheritance calculations are either exponential in the number of individuals or exponential in the number of loci [61]. There have been numerous and notable attempts to increase the speed of these calculations [85, 1, 29, 13, 32, 63, 23]. Recent work from statistics has focused on fast and efficient calculations of disease-linkage that avoid the full inheritance calculations [9, 96].

Statistical calculations on pedigrees are the principal method behind the most accurate disease-association approaches [80, 96]. In those approaches, the aim is to find the regions of the genome that are associated with the presence or absence of a disease among related individuals.

This chapter introduces pedigrees, inheritance probabilities, and likelihood calculations. Three probabilistic and combinatorial quantities of interest on pedigrees are introduced. The previously known hardness of these three problems given genotype data is discussed and a novel hardness result for the same problems given haplotype data is introduced.

3.1 Introduction

3.1.1 Pedigrees

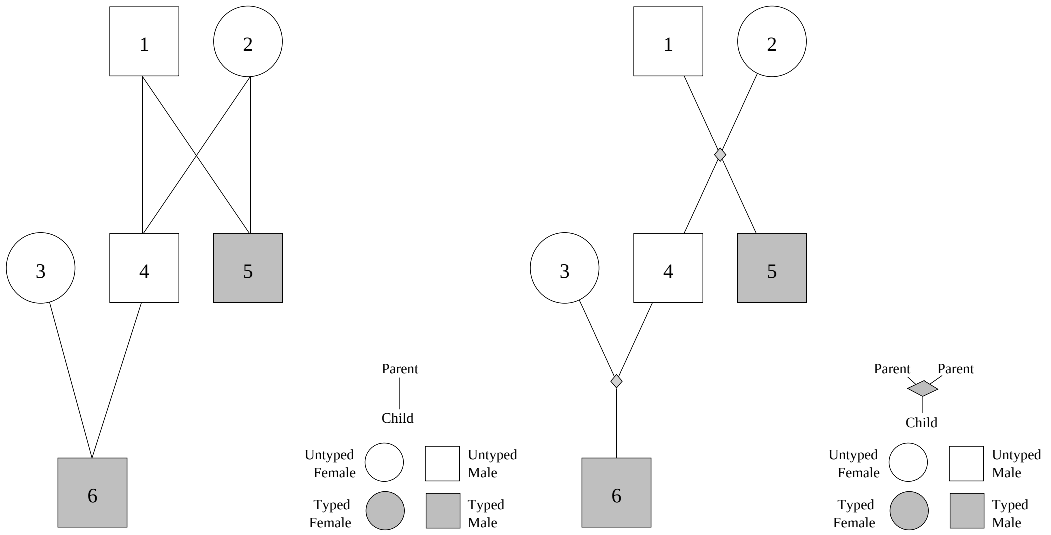


Figure 3.1: **Two Drawings of the Same Pedigree.** These two drawings are of the same pedigree graph. A third, more traditional, representation is shown in Figure 3.2

A *pedigree* is a directed acyclic graph where the set of nodes, I, are individuals, and directed edges indicate genetic inheritance between parent and child. A diploid pedigree (i.e. for humans) necessarily has either zero or two incoming edges for each person. The set, F, of individuals without incoming edges are referred to as pedigree *founders*. An individual, i, with two parents is a *non-founder*, and we will refer to their two parents as $m(i)$ and $p(i)$. Due to every person having a male and female parent, it is well known that legal pedigree graphs are four-colorable [94].

Pedigree graphs are drawn with the edges implicitly directed down from parent to child, without drawing the actual direction arrow on the edge. Circle nodes are females, boxes are males, as in Figure 3.1. There are three ways to draw the parent-child edges. The left panel of Figure 3.1 has one edge drawn for each parent-child pair. The right panel of Figure 3.1 bundles the edges from a pair of parents to their children into the diamond 'marriage' node. Figure 3.2 shows the traditional way that epidemiologists draw pedigrees where a horizontal edge between a male and female indicate marriage and an edge proceeds downward from the marriage line to indicate all the offspring of the coupling, so that full-siblings are adjacent to each other. At various points in this thesis we will use different representations based on convenience.

For convenience, we will number the generations backwards in time, with larger numbers being older generations. Let g be the number of generations of individuals in the graph. For example, if $g = 1$, then we are discussing only the *extant* individuals, those individuals

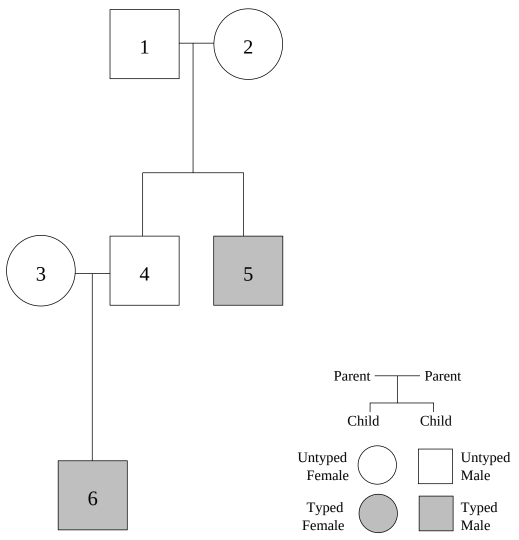


Figure 3.2: **Traditional Drawing of a Pedigree.** This is the same pedigree as that shown in Figure 3.1

at the most recent generation. If $g = 2$ the graph contains the extant individuals and their parents.

We call a pedigree *regular* when all individuals only mate with other individuals at the same generation. A pedigree is *monogamous* if and only if every individual mates with at most one other individual, so that there are no half-siblings. A pedigree has *inbreeding* if there exists two individuals, one the ancestor of the other, such that there are multiple paths in the pedigree graph connecting those two individuals.

Unlike a phylogeny, where the principal process is mutation, in a pedigree, the principal process is recombination. In pedigrees the typical assumption is that there is no mutation of alleles during inheritance. While this assumption is clearly not biologically accurate, genotyping errors are too common to be able to distinguish between actual mutations and genotyping errors.

Recombination along the genome is typically modeled as a Poisson process, where the distance between recombination breakpoints is drawn from an exponential distribution. The mean of the exponential is a function of the recombination rate [24, 7]. This is a model for recombination without interference, where interference means that the presence of one recombination breakpoint suppresses the occurrence of breakpoints in neighboring regions of the sequence [65].

3.1.2 Inheritance States and Identity by Descent (IBD)

The pedigree graph represents the possible inheritance options, i.e. which alleles can be copies of each other. Let us consider one position, j, in the genome. If there are $|I \setminus F|$ non-founders, we can represent the *inheritance state* of all alleles in the pedigree with a

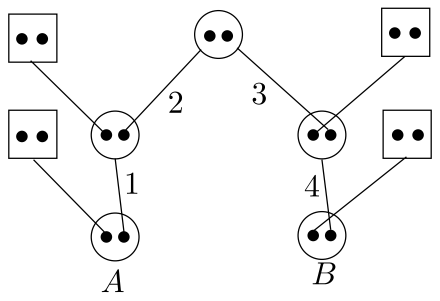


Figure 3.3: **Inheritance Options for Half-Cousin Pedigree.** The black disks are alleles, with each diploid person having two alleles. The inheritance edges indicate transmission of a parent's allele to the child. The four edges relevant to IBD inheritance for individuals A and B are numbered $1, ..., 4$. The alleles of each person are sorted so that the left allele is the one inherited from the father and the right allele is inherited from the mother. Each allele has a binary inheritance choice, in that it can be a copy of either of the parent's two alleles.

$|I \setminus F|$-bit vector s_j. For each non-founder, let us indicate the source of each maternal allele using the binary variable $s^m_{i,j} \in \{0, 1\}$, where the value 1 indicates that $x^m_{i,j}$ allele has grand-maternal origin and 0 indicates grand-paternal origin. Similarly, we define $s^p_{i,j}$ for the origin of i's paternal allele. The vector $s_j = \{s^p_{i,j}, s^m_{i,j} | \forall i\}$ is the inheritance state for locus j.

Individuals of interest are called *identical by descent (IBD)* if a particular founder allele was copied to each of the individuals. In general, the inheritance state is a collection of trees, since each allele is copied from a single parent.

For example, see Figure 3.3 where a half-cousin pedigree is shown in detail with the black disks representing alleles. The inheritance edges relevant to IBD relationships between individuals A and B are numbered $1, ..., 4$. Since there is one bit per edge, there are 16 inheritance states for the four numbered edges in this pedigree. Shown in the left panel of Figure 3.4, a non-IBD inheritance state for individuals A and B is 1011. There are only two inheritance states which yield IBD relationships, 1001 and 1111, the former is illustrated in the right panel of Figure 3.4.

3.1.3 Inheritance Probabilities

We represent a single chromosome as an ordered sequence of variables, x_j, where each variable takes on an *allele* value in $\{1, ..., k_j\}$. Each variable represents a *polymorphic site*, j, in the genome, where there are k_j possible sequence variants. Notice that even though we restricted ourselves to binary alleles in Chapter 2, we are now considering the general case. Since diploid individuals have two copies of each chromosome, one copy inherited from each parent, we will use a superscript m and p to indicate the maternal and paternal chromosomes respectively. For a particular individual i, the information on both copies of a particular chromosome at site j is represented as $x^m_{i,j}$ and $x^p_{i,j}$.

Furthermore, we assume that inheritance in the pedigree proceeds with recombination

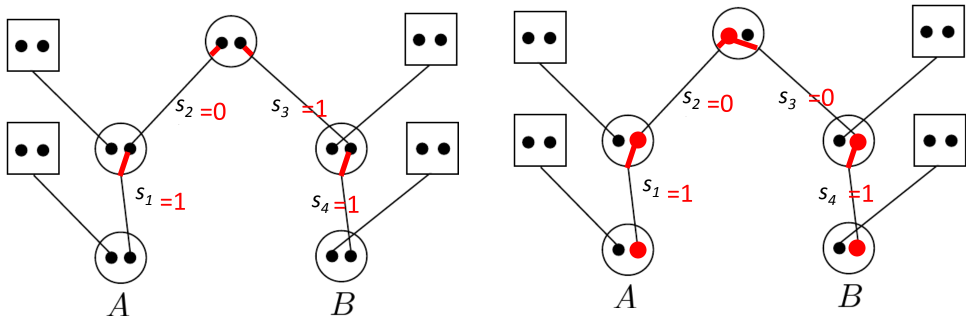


Figure 3.4: **Non-IBD and IBD Inheritance States.** For the pedigree from Figure 3.3, there are 16 possible inheritance states on the 4 relevant edges. Two inheritance states are shown here. The left panel shows a non-IBD inheritance state, 1011, where no allele from individuals A and B is copied from the same ancestor. The right panel shows an IBD inheritance state, 1001, where the red alleles in individuals A and B are copied from the same ancestor. Indeed, there is only one other IBD inheritance state, 1111, where the right allele of the grandmother is copied to individuals A and B.

and without mutation (i.e. Mendelian inheritance at each site). This imposes consistency rules on parents and children: the allele $x^m_{i,j}$ must appear in the mother $m(i)$'s genome as either the grand-maternal or grand-paternal allele, $x^m_{m(i),j}$ or $x^p_{m(i),j}$, and similarly for the paternal allele and the father $p(i)$'s genome.

Let x be a vector containing all the haplotypes x^m_i, x^p_i for all individuals $i \in I$, then we are interested in the probability

$$\mathbb{P}[x] = \prod_{f \in F} \mathbb{P}[x^p_f]\mathbb{P}[x^m_f] \prod_{i \in I \setminus F} \mathbb{P}[x^p_i | x^p_{p(i)}, x^m_{p(i)}]\mathbb{P}[x^m_i | x^p_{m(i)}, x^m_{m(i)}], \tag{3.1}$$

where the superscript m and p indicate maternal and paternal alleles, while the functions $m(i)$ and $p(i)$ indicate parents of i. The first product is over the independent founder individuals whose haplotypes are drawn from a prior distribution which is often the uniform distribution. The second product, over the non-founders, contains the probabilities for the children to inherit their haplotypes from their parents. The unobserved vector x is not immediately derived from observed haplotype data, since vector x contains haplotype alleles labeled with their parental origins for all the individuals. To compute this quantity, we need notation to represent the parental origins of each allele where differing origins for neighboring haplotype alleles will indicate recombination events.

Recall that s_j is a binary vector giving the inheritance state for site j. A recombination is observed at consecutive sites as a change in the binary value of a source vector, for instance, $s^m_{i,j} = p$ and $s^m_{i,j+1} = m$. To compute the inheritance portion of Equation 3.1, we will sum over the inheritance options $\mathbb{P}[x] = \sum_s \mathbb{P}[x|s]\mathbb{P}[s]$ where $\mathbb{P}[s] = 1/2^{2|I \setminus F|}$

As introduced in Chapter 1, we can observe two kinds of data for pedigree individuals whose genetic material is available. The first, and most common, is genotype data, a tuple of alleles $(g^0_{i,j}, g^1_{i,j})$ that must appear in the variables $x^m_{i,j}$ and $x^p_{i,j}$ for each site j.

Since these alleles are unlabeled for origin, we do not know which allele was inherited from which parent. The second type of data is haplotypes, where we observe two sequences of alleles h_i^0 and h_i^1 and each sequence represents alleles that were inherited together from the same parent. However, we do not know which sequence is maternal and which is paternal. Let $\chi(s_{i,j}^k) = p$ if inheritance bit $s_{i,j}^k = 0$ and $\chi(s_{i,j}^k) = m$ if inheritance bit $s_{i,j}^k = 1$. For either type of data, define a function $C_{i,j}$ for locus j which indicates compatibility of the assigned haplotype alleles with the data and requires inheritance consistency between generations. Specifically, for genotype data $C_{i,j} = 1$ if $x_{i,j}^m = x_{m(i),j}^{\chi(s_{i,j}^m)}$, $x_{i,j}^p = x_{f(i),j}^{\chi(s_{i,j}^p)}$, and $\{x_{i,j}^m, x_{i,j}^p\} = \{g_{i,j}^0, g_{i,j}^1\}$. Under haplotype data, the $C_{i,j} = 1$ when the first two equalities, above, hold and $\{x_{i,j}^m, x_{i,j}^p\} = \{h_{i,j}^0, h_{i,j}^1\}$, which are the haplotype alleles at locus j.

Now, we write Equation 3.1 as a function of the per-site recombination probability $\theta \leq 0.5$. For particular values of all the haplotype alleles $x_{i,j}^m$ and $x_{i,j}^p$, the haplotype probability conditional on the inheritance options and the observed data through $C_{i,j}$ is

$$\mathbb{P}[x|s] = \prod_{f \in F} \prod_{j=1}^{l} C_{f,j} \mathbb{P}[x_{f,j}^p] \mathbb{P}[x_{f,j}^m] \prod_{i \in I \setminus F} C_{i,1} \prod_{j=2}^{l} C_{i,j} \cdot \theta^{(R_{i,j}^m + R_{i,j}^p)} \cdot (1-\theta)^{(2 - R_{i,j}^m - R_{i,j}^p)}$$

where $R_{i,j}^m = \mathbb{I}[s_{i,j-1}^m \neq s_{i,j}^m]$ and $R_{i,j}^p = \mathbb{I}[s_{i,j-1}^p \neq s_{i,j}^p]$. Figure 3.5 illustrates each component of $\mathbb{P}[x|s]$ as well as $\mathbb{P}[s]$.

3.2 Problems of Interest

Given a pedigree and some observed genotype or haplotype data, there are three problem formulations that we might be interested in. The first is to compute the probability of some observed data, while the last two problems find values for the unobserved haplotypes of individuals in the pedigree.

Likelihood. Find the probability of the observed data by summing over all the possible unobserved haplotypes, i.e.

$$\sum_x \sum_s \mathbb{P}[x|s]\mathbb{P}[s].$$

Maximum Probability. Find the values of $x_{i,j}^m$ and $x_{i,j}^p$ that maximize the probability of the data, i.e.

$$\max_x \sum_s \mathbb{P}[x|s]\mathbb{P}[s].$$

Minimum Recombination. Find the values of $x_{i,j}^m$ and $x_{i,j}^p$ that minimize the number of required recombinations, i.e.

$$\min_{x,s} \sum_i \sum_{j=2}^{l} \mathbb{I}[s_{i,j-1}^p \neq s_{i,j}^p] + \mathbb{I}[s_{i,j-1}^m \neq s_{i,j}^m]. \tag{3.2}$$

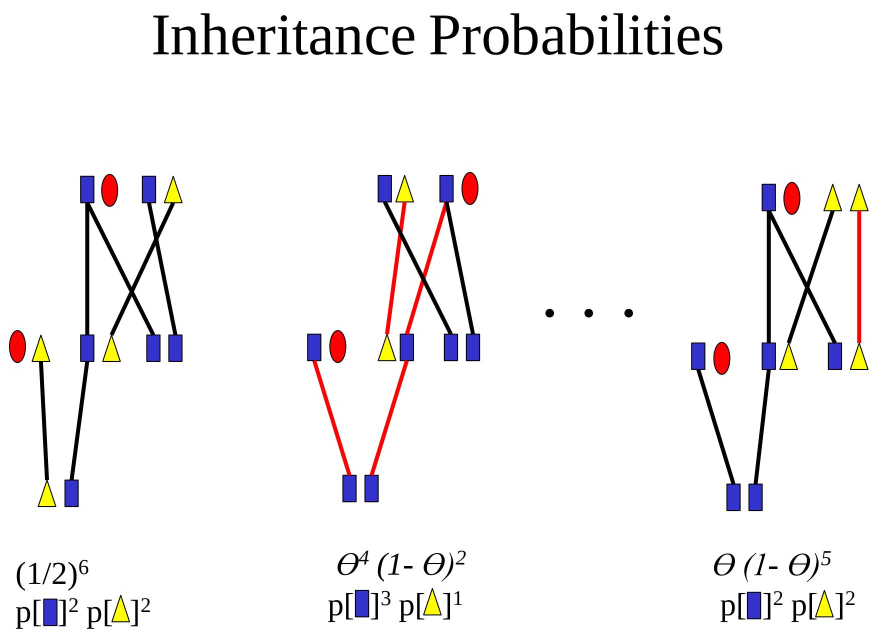


Figure 3.5: **Illustration of Inheritance Probabilities.** The pedigree shown in this figure is the same as that given in Figure 3.1. Each pair of colored polygons are the two alleles for a single individual. Three positions in the genome are illustrated as three separate inheritance state graphs. Edges connect alleles that are copies of each other. The contributions to the inheritance probability for each position are recorded below the graph. For the first locus, $\mathbb{P}[s] = 1/2^6$ and the founder contributions are given. For the second and last locus, the recombination contributions and founder contributions are given. For recombination contributions, we have θ to the power of the number of recombination edges, shown in red. For the founder contribution, we multiply the probabilities of each specified founder allele. The red founder alleles are unspecified and contribute probability 1.

The likelihood is commonly used for estimating site-specific recombination rates, relationship testing, computing p-values for association tests, and performing linkage analysis, which is a likelihood ratio test for the recombination rate between a hypothetical disease locus and the observed genetic loci. Haplotype and/or IBD inferences, obtained by maximizing the probability or minimizing the recombinations, are useful for non-parametric association tests, tests on haplotypes, and tests where there is disease information for unobserved genomes.

The pedigree likelihood was introduced long before there were computers to do the calculations. The two algorithms most commonly used for computing these likelihoods are the Elston-Stewart algorithm [27] introduced in 1971 and the Lander-Green algorithm [59] introduced in 1987. The former algorithm is exponential in the number of sites, while the later is exponential in the number of individuals. More recently these calculations have been formulated using the graphical model framework from machine learning [61]. We now know that both the Elston-Stewart and Lander-Green algorithms deal with the same pedigree graphical model, they just choose a different elimination order for the sum-product algorithm. The Elston-Stewart algorithm eliminates individuals in a process called 'peeling', while the Lander-Green algorithm eliminates loci.

3.2.1 The Peeling Algorithm and Elston-Stewart

Elston and Stewart's peeling algorithm [27] is usually applied to outbred pedigrees. This requirement is introduced to remove loops from the pedigree graph, i.e. removing the possibility of inbreeding. Loops are troublesome for the peeling algorithm, because they increase the tree-width of the pedigree graph, thereby increasing the sizes of the cliques required by the junction-tree algorithm [61, 58], a generalization of the sum-product algorithm to non-tree-like graphs. Since, the clique sizes directly influence the running time, it is best to keep them small.

For the purposes of this section, we will consider monogamous outbred pedigrees which consist of a single lineage denoted $L(p, q)$, i.e. the descendants of a monogamous founding pair (MFP), p and q, and the founder parents of the descendants of p, q. The Elston-Stewart algorithm is a dynamic programming algorithm that proceeds up the pedigree from the leaves.

Let indices i and j denote haplotypes. Let L_r be the subpedigree rooted at a child r of the founders. For each node w in L_r let $T(w)$ be the subtree consisting of individual w and all its descendants; in particular, $T(r) = L_r$. If $w \neq r$ let $a(w)$ denote the haplotype that w receives from its founder parent, and $b(w)$, the haplotype that w receives from its non-founder parent in $L(p, q)$, and let $\alpha(i)$ be the probability that $a(w) = i$. Let $a(r)$ and $b(r)$ the two lineage haplotypes of r (all relevant properties are unaffected by interchanging $a(r)$ and $b(r)$). Let $C(w, i, j)$ be an indicator variable that is 1 if the haplotype pair (i, j) is consistent with the genotype of individual w. Let $H(w)$ denote the set of children of w.

We want to compute:

$$\mathbb{P}_r(i,j) = \sum_{b(x),a(x)\forall x\in T(r)} C(r,i,j)Pr[a(r)=i,b(r)=j|\text{MFP haplotypes}] \cdot \prod_{x\in T(r)} Pr[b(x),a(x)|b(l(x)),a(l(x)),h^0_{n(x)},h^1_{n(x)}]$$

where $n(x)$ is the founder parent of individual x. If we re-arrange the equation, we get a message for each individual just as in the sum-product algorithm. The message is a factor in the complete marginal. Then for leaf-nodes u, and internal nodes w, where $w \neq r$ we have the recursive equations:

Leaf:

$$\mathbb{P}_u(i) = \sum_j \alpha(j)C(u,i,j)$$

Internal Node:

$$\mathbb{P}_w(i) = \sum_j \alpha(j)C(w,i,j) \prod_{x\in H(w)} \frac{\mathbb{P}_x(i)+\mathbb{P}_x(j)}{2}$$

Founder Child:

$$\mathbb{P}_r(i,j) = C(r,i,j) \prod_{x\in H(r)} \frac{\mathbb{P}_x(i)+\mathbb{P}_x(j)}{2}$$

We can compute all these quantities by working upward from the leaves of the tree L_r to the root. The running time of the algorithm is $O(N^2L)$ where N is the number of haplotypes and L is the number of individuals in L_r.

We now extend the restricted model to encompass the monogamous pair of founders, p and q in their lineage $L(p,q)$. Extending the restricted model to this case, we make the same assumptions as above except that the MFP-children $r \in H(p)$ have two lineage haplotypes. We continue to assume that each founder parent of a child in $L(p,q)$ is the parent of at most one node in $L(p,q)$ and that a probability distribution α independently determines the probability of haplotype transmission for haplotypes from the founders other than p,q.

Let $\mathbb{P}_{p,q}(i,j,k,l)$ be the probability that the haplotypes of all nodes of $L(p,q)$ are compatible with the genotype data at those nodes, given that p has haplotypes i and j and q has haplotypes k and l. Then the recursive probabilities for the **founder pair** are

$$\mathbb{P}_{p,q}(i,j,k,l) = C(p,i,j)C(q,k,l) \prod_{r\in H(p)} \frac{\mathbb{P}_r(i,k)+\mathbb{P}_r(i,l)+\mathbb{P}_r(j,k)+\mathbb{P}_r(j,l)}{4}$$

where, for each $r \in H(p)$, the quantities on the right-hand side are computed according to the recursive algorithm given above. The execution time of the computation is $O(N^4L)$ where $N = 2^m$ is the number of haplotypes, m is the number of SNPs, and L is the number of individuals in the lineage.

3.2.2 Hidden Markov Models, Lander-Green, and the Forward-Backward Algorithm

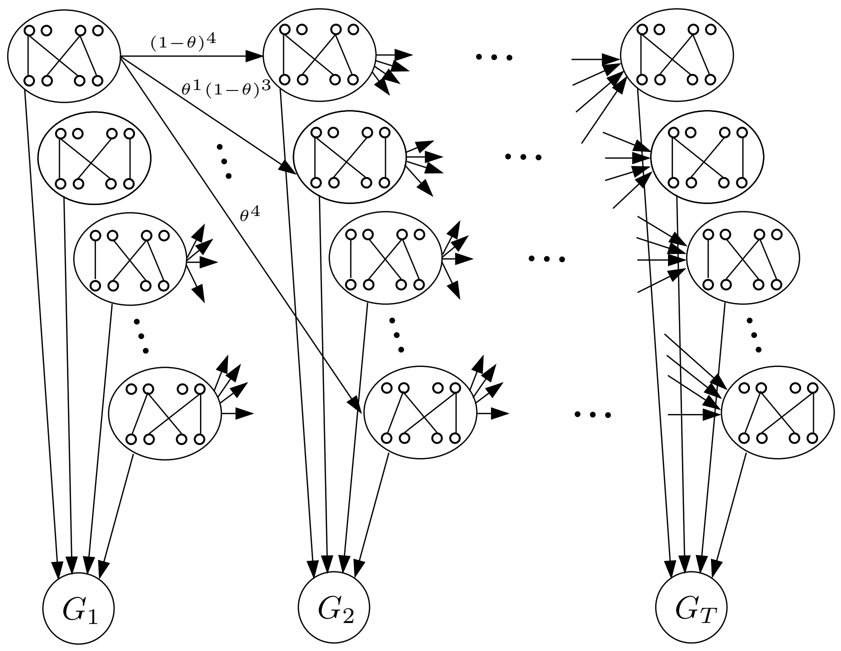


Figure 3.6: **Lander-Green Hidden Markov Model.** The emission states are labeled $G_1, ..., G_T$. Example hidden states for a four-person pedigree are shown above the emission states. The alleles of each person in the pedigree are drawn as a hollow disk, and the alleles are implicitly grouped in pairs. There are an exponential number of hidden states, one state for each inheritance vector. Recombination probabilities give the transition probabilities for the HMM. If two b-bit inheritance vectors differ at i bits, then the transition probability is $\theta_{j-1}^i(1-\theta_{j-1})^{m-i}$.

The pedigree likelihood can obtained by constructing a hidden Markov model for the linkage dependencies along the genome. At each locus, the HMM considers the constraints given by the genotype data. We first use the forward-backward algorithm [79] to compute the marginal inheritance probabilities for each locus using a hidden Markov model. While the forward-backward algorithm is the terminology used for general HMMs, the pedigree algorithm has a special name, the Lander-Green algorithm [59].

The likelihood can be modeled using a hidden Markov model along the genome with inheritance paths as hidden states. The transition probabilities are functions of θ_j and the number of recombinations between a given pair of inheritance graphs. Let $b = |I \setminus F|$ be the number of non-founders. Let X_j be the random variable for the hidden state at position j. Then the probability of transitioning from inheritance state s_{j-1} to s_j is

$$Pr[X_j = s_j | X_{j-1} = s_{j-1}] = \theta_{j-1}^{|s_j - s_{j-1}|}(1-\theta_{j-1})^{b-|s_j-s_{j-1}|}.$$

where θ_{j-1} is the recombination rate between site $j-1$ and site j. See Figure 3.6 for an illustration of the HMM for a four-person pedigree.

Given the data, we compute the marginal inheritance path probabilities at each site by using the forward-backward algorithm for HMMs. Sobel and Lange [85] described a method for enumerating only the inheritance paths compatible with the allele data observed at each locus. There are at most $k = 2^{2|I \setminus F|}$ inheritance paths when $I \setminus F$ is the set of non-founder individuals, and both the forward and backward recursions do an $O(k^2)$ calculation at each site, making this algorithm exponential in the number of individuals.

3.3 Hardness

The work in this section was a novel contribution by the author [50, 48].

Single-molecule sequencing is an attractive alternative to genotyping and could yield haplotypes for individuals in a pedigree [26]. In order to exploit the information contained in haplotype data, we need to understand the instances where diploid inheritance is computationally tractable given haplotype data.

Pedigree analysis with genotype data is well studied in terms of complexity [76, 62] and algorithms [27, 59, 85]. Less is known about haplotype data on pedigrees. This section considers the three probabilistic and combinatorial problems introduced earlier and introduces novel hardness results for haplotype data. Given haplotype data on a pedigree, finding both minimum recombination and maximum probability haplotypes is as tractable as computing the same quantities for pedigrees with genotype data (i.e., these problems are NP- and #P-hard, respectively).

With genotype data, the likelihood and minimum recombination problems are NP-hard, while the maximum probability problem is #P-hard. Piccolboni and Gusfield [76] proved the hardness of the likelihood and maximum probability computations by relying on a single locus sub-pedigree with half-siblings. Although their paper discussed a more elaborate setting involving a phenotype, their proof, however, applies to this setting. Li and Jiang proved the minimum recombination problem to be hard by using a two-locus sub-pedigree with half-siblings [62]. In all these proofs, half-siblings were pivotal to establishing reductions from well known NP and #P problems.

In this section, we introduce a simple and powerful reduction that converts any genotype problem on a pedigree of n individuals into a haplotype problem on a pedigree of at most $6n$ individuals. This reduction is simple, because it merely introduces four full-siblings and an extra parent for each genotyped individual. We do not need complicated structures involving inbreeding or half-siblings. The reduction works equally well for all three problem formulations. Note that the proofs in this section require that per-site recombination rate $\theta_j = \theta$ is equal across all sites.

Mapping. Given a pedigree with genotype data, for any of the three pedigree problems, we define a polynomial mapping to a corresponding haplotype problem with exactly $5|G|$ individuals haplotyped. First we create the pedigree graph for the new haplotype instance, and later we construct the required haplotype observations from the genotype data.

Let $G \subset I$ represent the set of genotyped individuals in a pedigree having individuals I and edges E. We will create a haplotype instance of the problem, with individuals $H \cup I$

and edges $R \cup E$. To obtain the set H, we add five individuals, i_0, i_1, i_2, i_3, i_4, to H for every individual $i \in G$. The set of new relationship edges, R, will connect individuals in sets H and G. Specifically, the edges stipulate that i and i_0 are the parents of full-siblings i_1, i_2, i_3, and i_4 by including the edges: $i_0 \to i_1$, $i_0 \to i_2$, $i_0 \to i_3$, $i_0 \to i_4$, $i \to i_1$, $i \to i_2$, $i \to i_3$, and $i \to i_4$. We will refer to these five individuals, i_0, i_1, i_2, i_3, and i_4, and their relationships with i as the *proxy family* for individual i. For example in Figure 3.7, the 6-individual genotype pedigree in becomes a 21-individual haplotype pedigree. This produces a pedigree graph with exactly $5|G| + |I|$ individuals and $8|G| + |E|$ edges.

To obtain the new haplotype data from the genotype data, we type only individuals in H such that the corresponding genotyped individual in G is required, by the rules of inheritance, to have the observed genotypes. Without loss of generality, assume that the genotype alleles are sorted such that $g^0_{i,j} < g^1_{i,j}$. Now we can easily constrain the parental genotype for individual $i \in G$ by giving the spouse, i_0, homozygous haplotypes of all ones while giving child i_1 the haplotypes $\{\vec{1}, g^0_i\}$, child i_2 haplotypes $\{\vec{1}, g^1_i\}$. This guarantees the correct genotype, but does not ensure that the haplotypes of that genotype have the same probability or number of recombinations.

Since there is an arbitrary assorting of genotype alleles at neighboring loci into the parent haplotypes x^p_i and x^m_i, we will use the remaining two children to represent possible re-assortments of the genotyped parent's T_i heterozygous loci, indexed by t_j where $1 \leq j \leq T_i$. In addition to the haplotype $\vec{1}$, child i_3, will have haplotype consisting of $h_{i_3,t_j} := g^{1-j \bmod 2}_{i,t_j}$ while child i_4 has the genotyped parent's complementary alleles $h_{i_4,t_j} := g^{j \bmod 2}_{i,t_j}$. This results in child i_3 and i_4 alternating in having the smaller allele at every other heterozygous locus.

This reduction preserves the solutions to the three problems up to constant factors or constant coefficients. Specifically, the solution to the haplotype version of the problem is the solution to the genotype version with the values of the functions being related by constant factors or coefficients, depending on whether the function is a recombination count or a probability.

Lemma 3.3.1. *Let r_g be the minimum number of recombinations in the genotype problem instance. The mapping yields a haplotype problem instance having $r_h = r_g + \sum_{i \in G} 2(T_i - 1)$ for the minimum number of recombinations, where T_i is the number of heterozygous sites in genotype i.*

To prove this result, we exploit the alternating pattern of alleles assigned to the four children. This pattern forces there to be two recombinations, among the four children, between consecutive heterozygous loci.

Proof of Constant Dependence for Minimum Recombination. Consider the haplotype instance of the problem. Recall that set G is defined as the individuals who are genotyped in the genotype problem instance, and, by construction, they are not haplotyped in the haplotype problem instance. For each $i \in G$ the rules of inheritance applied to i's proxy family dictate that the set of alleles at each position are given by $g^0_{i,j}$ and $g^1_{i,j}$. Therefore, the proxy family dictates the genotype of i.

Since the haplotypes for all the typed individuals are completely given, we only need to consider the assortment of the alleles from g^0_i and g^1_i into the maternal and paternal

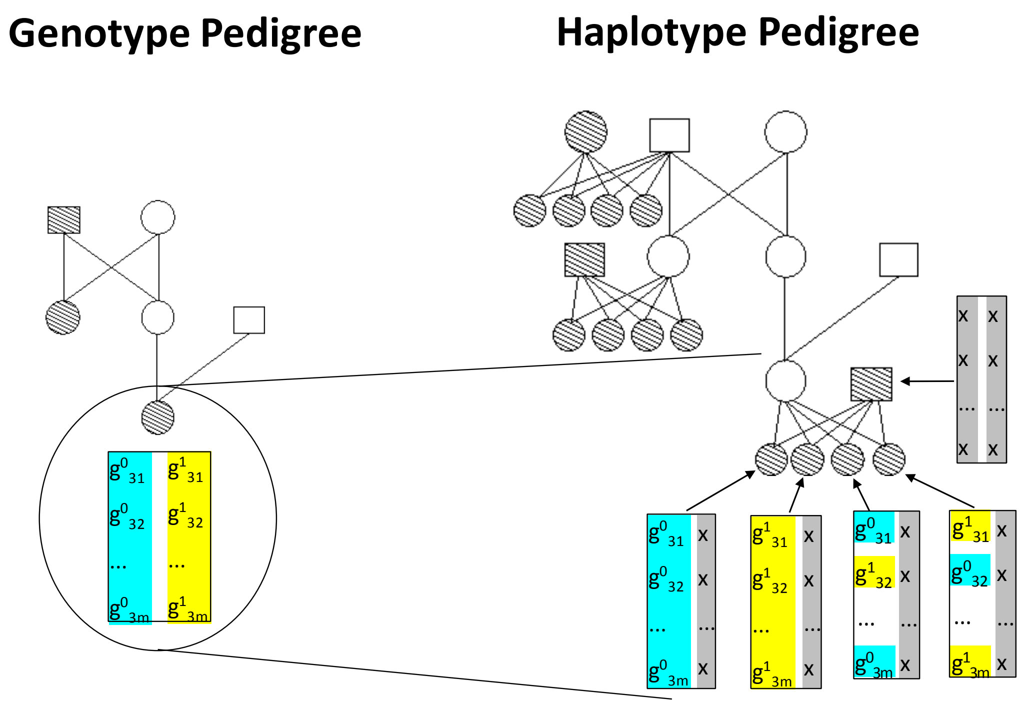


Figure 3.7: **Genotype and Haplotype Pedigrees.** (Left) Genotyped individuals are shaded. We show one typed individual's genotype which is expanded into the haplotypes of the typed individuals in the proxy family of the haplotype pedigree. (Right) Haplotyped individuals are shaded. For each of the genotyped individuals in the left panel, the mapping adds a nuclear family containing five new individuals with haplotypes having the pattern illustrated.

alleles of individual i. Clearly this assortment determines the number of recombinations that the proxy family contributes to Equation (3.2). However, we will use induction along the genome to show that every possible phasing of the parental genotype induces the same minimum number of recombinations among the four children, namely $2(T_i - 1)$.

Now we define an arbitrary assortment of the genotype alleles into two haplotypes for person i. We can think of this parental genotype for l loci as a string $s \in \{H, T\}^l$, where H represents a homozygous site and T a heterozygous site. Recall that T_i is the number of heterozygous sites in the genotype string, and those sites appear at indices t_j where $1 \leq j \leq T_i$. For this genotype there are $2^{T_i - 1}$ pairs of haplotypes that phase the given genotype. Represent each pair by setting $T_i - 1$ binary variables

$$P_{t_j} = \begin{cases} 0, & \text{if } x^p_{i,t_j} < x^m_{i,t_j}, \\ 1, & \text{otherwise.} \end{cases}$$

Note, that we are only interested in the origin of the children's haplotypes, rather than in the origin of i's haplotypes, so the p and m can arbitrarily label either haplotype.

Since $\{i_1, i_2\}$ between them have the parent genotype at every locus, one of them has origin p while the other has origin m, and similarly for $\{i_3, i_4\}$. For each locus, indicate the paternal origin of the allele for individuals i_1 and i_3, respectively with Q_j and S_j. Formally, $Q_j = 1$ if both $h_{i_1,j} = x^p_{i,j}$ and $h_{i_2,j} = x^m_{i,j}$ while $Q_j = 0$ otherwise. Similarly, $S_j = 1$ if both $h_{i_3,j} = x^p_{i,j}$ and $h_{i_4,j} = x^m_{i,j}$ while $Q_j = 0$ otherwise.

Define R_j as the minimum recombination count before locus j. Notice that P_{t_1} sets the origin of all the child haplotypes, therefore $R_{t_1} = 0$, since all preceding homozygous loci can have the same origin as locus t_1.

From t_j to t_{j+1} we have two cases:

1. If $P_{t_j} = P_{t_{j+1}}$, then $Q_{t_j} = Q_{t_{j+1}}$ and $S_{t_j} \neq S_{t_{j+1}}$, by the alternating construction of children i_3 and i_4 as compared with i_1 and i_2.

2. Similarly, if $P_{t_j} \neq P_{t_{j+1}}$, then $Q_{t_j} \neq Q_{t_{j+1}}$ and $S_{t_j} = S_{t_{j+1}}$.

Furthermore, regardless of the number of homozygous loci separating t_j and t_{j+1}, the number of recombinations can only be increased. Therefore, we have the recursion

$$R_{t_{j+1}} = 2 + R_{t_j},$$

proving the lemma. □

After applying the mapping, the haplotype probability turns out to have a coefficient that is independent of the haplotype assignment to the non-founding parent of the proxy family. This coefficient can be computed in linear time from the genotype data using a Markov chain. The Markov chain has 16 states and has a transition step between each pair of neighboring loci. This small Markov model can be thought of in the sum-product algorithm as an elimination of the typed individuals in the proxy family; alternatively, it is also equivalent to peeling-off the typed proxy individuals in the Elston-Stewart algorithm [27]. Once we have this coefficient, independent of the haplotype assignment, it is clear that the likelihood and maximum probability haplotype problems also have haplotype solutions related proportionally to the genotype solution.

Lemma 3.3.2. *The mapping yields a haplotype problem instance having haplotype probabilities proportional to the haplotype probabilities of the genotype instance. Specifically, for all x,*

$$\mathbb{P}_h[x] \;=\; \left(\mathbb{P}_g\big[\{x_i | i \in I\}\big]\right) \prod_{i \in G} p_t(i) \prod_j \mathbb{P}[x^p_{i_0,j} = 1]\mathbb{P}[x^m_{i_0,j} = 1]$$

where the proxy family transmission probability is a function of genotype g_i, the recombination rate $\theta \leq 0.5$, and of the transition matrices P, Q_{0110}, and Q_{1001},

$$p_t(i) = \left(\frac{1}{16}\right) \vec{1} \cdot P^{h_0} \prod_{j=0}^{T_i} \big(O_j Q_{0110} + (1 - O_j) Q_{1001}\big) \cdot P^{h_j} \cdot \vec{1}^T$$

and O_j indicates whether index j is odd, h_0 is the number of homozygous loci that begin proxy parent's genotype, and h_j is the number of consecutive homozygous loci after the j'th heterozygous locus where there are T_i heterozygous loci for proxy parent i. The transition probabilities are given by $P_{ij} = \theta^{H(i,j)}(1-\theta)^{4-H(i,j)}$ where $H(i,j)$ is the Hamming distance between inheritance states i and j. Let Q_{0110} be a transition matrix having non-zero recombination probabilities only in column 0110 *(i.e. $Q_{0110,i,j} = P_{ij}$ when $j = 0110$). Similarly, let Q_{1001} be a transition matrix with non-zero recombination probabilities only in column* 1001.

Proof of Proportional Haplotype Probability. Without loss of generality, assume that individuals $i \in G$ are all fathers in their proxy family. This is simply for convenience of notation.

Let x be any fixed assignment of haplotypes to all the individuals in the pedigree. When conditioning on the assigned haplotypes for individual i, the probability of the proxy family of i is independent of the probability for the rest of the pedigree. Since we can say this for all the proxy families, the terms in the probability for the pedigree individuals in set I (i.e. those also in the genotype pedigree) are equal to the probability on the genotype data in the genotype pedigree. Therefore, we write that

$$\begin{aligned}\sum_s \mathbb{P}_h[x|s]\mathbb{P}[s] \;=\; & \sum_s \mathbb{P}_g\big[\{x_i|i \in I\}|\{s_i|i \in I\}\big]\ \mathbb{P}\big[\{s_i \in I\}\big] \prod_{i\in G}\left(\prod_j \mathbb{P}[x^p_{i_0,j} = 1]\mathbb{P}[x^m_{i_0,j} = 1]\right) \\ & \cdot \left(\prod_k \mathbb{P}[x^p_{i_k}|x^p_{f(i_k)}, x^m_{f(i_k)}, s^p_{i_k}]\mathbb{P}[x^m_{i_k}|x^p_{m(i_k)}, x^m_{m(i_k)}, s^m_{i_k}]\mathbb{P}[s^p_{i_k}]\mathbb{P}[s^m_{i_k}]\right).\end{aligned}$$

The sum over vector s can be split into sums over the component pieces. The sums involving the s_{i_k} can be distributed into the product over k, since that is the only place they are used. Let $s_{i_k} = (s^p_{i_k}, s^m_{i_k})$. We easily see that $\mathbb{P}[x^m_{i_k}|x^p_{m(i_k)}, x^m_{m(i_k)}, s^m_{i_k}]\mathbb{P}[s^m_{i_k}] = 1$, since there are two ways to inherit the 1-allele from the mother, and all of them are compatible.

$$\begin{aligned}\sum_s \mathbb{P}_h[x|s]\mathbb{P}[s] \;=\; & \sum_{\{s_i|i\in I\}} \mathbb{P}_g\big[\{x_i|i \in I\}|\{s_i|i \in I\}\big]\ \mathbb{P}\big[\{s_i \in I\}\big] \prod_{i\in G}\left(\prod_j \mathbb{P}[x^p_{i_0,j} = 1]\mathbb{P}[x^m_{i_0,j} = 1]\right) \\ & \cdot \left(\prod_k \sum_{s_{i_k}} \mathbb{P}[x^p_{i_k}|x^p_{f(i_k)}, x^m_{f(i_k)}, s^p_{i_k}]\mathbb{P}[s^p_{i_k}]\right).\end{aligned}$$

Let $p_t(i)$ be the transmission probability for the proxy family, defined as

$$p_t(i) = \prod_k \sum_{s_{i_k}} \mathbb{P}[x^p_{i_k} | x^p_{f(i_k)}, x^m_{f(i_k)}, s^p_{i_k}] \mathbb{P}[s^p_{i_k}].$$

View this probability as a Markov chain along the genome with a state space of size 2^4 where each state indicates the inheritance of $(s_{i_1}, s_{i_2}, s_{i_3}, s_{i_4})$. The transition probabilities are given by $P_{ij} = \theta^{H(i,j)}(1-\theta)^{4-H(i,j)}$ where $H(i,j)$ is the Hamming distance between inheritance states i and j. By design, the transitions allowed by the data have an unusual structure dictated by the heterozygous loci of the proxy parent. Specifically, at a heterozygous locus, there is exactly one inheritance state that satisfies the children's haplotypes. At homozygous loci, all the inheritance states are allowed. So, we compute this probability using the l-state transition probabilities to determine the contribution of a particular stretch of l homozygous loci that are followed by a heterozygous locus. Notice that the heterozygous locus has, as inheritance indicators, either $(0,1,1,0)$ or $(1,0,0,1)$, and these alternate between consecutive heterozygous loci.

Let Q_{0110} be a transition matrix having non-zero recombination probabilities only in column 0110 (i.e. $Q_{0110,i,j} = P_{ij}$ when $j = 0110$). Similarly, let Q_{1001} be a transition matrix with non-zero recombination probabilities only in column 1001. Let h_0 be the number of homozygous loci that begin proxy parent's genotype and let h_j be the number of consecutive homozygous loci after the j'th heterozygous locus where $1 \leq j \leq T_i$ and T_i is the number of heterozygous loci for proxy parent i. Now, we can write the transmission probability in terms of matrix operations

$$p_t(i) = \left(\frac{1}{16}\right) \vec{1} \cdot P^{h_0} \prod_{j=0}^{T_i} \big(Z_j Q_{0110} + (1-Z_j) Q_{1001}\big) \cdot P^{h_j} \cdot \vec{1}^T$$

where Z_j indicates whether the j'th heterozygous locus has inheritance indicators $(0,1,1,0)$. The column vector of ones at the end simply sums all final state probabilities to obtain the total probability.

Finally, notice that the two heterozygous inheritance states $(0,1,1,0)$ and $(1,0,0,1)$ are arbitrarily labeled. The main feature is that these states alternate at heterozygous loci, and it does not matter which one occurs first. So, we can write

$$p_t(i) = \left(\frac{1}{16}\right) \vec{1} \cdot P^{h_0} \prod_{j=0}^{T_i} \big(O_j Q_{0110} + (1-O_j) Q_{1001}\big) \cdot P^{h_j} \cdot \vec{1}^T$$

where O_j indicates the event that j is odd. Now we have a quantity that is a function of the genotype data and not dependent on the haplotypes under consideration.

$\square$

Corollary 3.3.1. *The mapping yields a haplotype problem instance having a likelihood and maximum probability proportional, respectively, to the likelihood and maximum probability* *of the genotype instance. Specifically,*

$$\sum_x \sum_s \mathbb{P}_h[x|s]\mathbb{P}[s] = \left(\sum_{\{x_i|i \in I\}} \sum_{\{s_i|i \in I\}} \mathbb{P}_g\big[\{x_i|i \in I\}|\{s_i|i \in I\}\big] \; \mathbb{P}\big[\{s_i \in I\}\big] \right) \cdot \prod_{i \in G} p_t(i) \prod_j \mathbb{P}[x^p_{i_0,j} = 1]\mathbb{P}[x^m_{i_0,j} = 1]$$

and

$$\max_x \sum_x \mathbb{P}_h[x] = \left(\max_{\{x_i|i \in I\}} \mathbb{P}_g[\{x_i|i \in I\}] \right) \cdot \prod_{i \in G} p_t(i) \prod_j \mathbb{P}[x^p_{i_0,j} = 1]\mathbb{P}[x^m_{i_0,j} = 1]$$

where $p_t(i)$ is proxy family i's transmission probability as defined in Lemma 3.3.2.

Proof of Proportional Likelihood and Maximum Probability. Lemma 3.3.2 shows that X is independent of the coefficient of proportionality between the haplotype probability and the genotype probability. Therefore, this coefficient factors out of both the likelihood and the maximum probability equations. $\square$

Although this reduction establishes the hardness of these haplotype pedigree problems, it does so by constructing children whose haplotypes require many recombinations and would be extremely unlikely to occur naturally. Accordingly, we suspect that realistic instances of these haplotyping problems may provide more information about the locations of recombinations than genotype instances.

Chapter 4

Algorithms for Inference

Statistical inference is the process of finding parameter values to fit a statistical model to data. Often inference is defined as an optimization problem. In this chapter, we make use of practical methods from machine learning including the sum-product algorithm and sampling algorithms.

The first problem considered in this chapter is to infer haplotypes from genotype data on pedigrees where the chosen haplotypes should maximize the inheritance likelihood. This is done without explicitly computing the likelihood by using a Gibbs sampler, see Section 4.1. We will see that that haplotype inference has applications to improving the accuracy of association studies on pedigrees.

The second problem is to infer recombination rates from haplotype data. This is done using an HMM similar to the Lander-Green algorithm described in Chapter 3 and the forward-backward algorithm for inference of the maximum-likelihood recombination rate. Simulation results show that the best recombination rate estimates are obtained from haplotype data where every individual in the pedigree is haplotyped.

The final algorithm in this chapter is a method for removing symmetries from the IBD hidden states of any pedigree HMM. This method reduces the number of hidden states, therefore improving the efficiency of sum-product calculations on the HMM, including the forward-backward algorithm. Simulations show that the exponential state-space reduction algorithm can decrease the number of hidden states exponentially.

4.1 Gibbs Sampler

The work in this section was performed as a collaboration between the author, Halperin, and Karp [55, 51].

In this section, we consider a special case of the pedigree haplotyping problem for complex pedigrees having multiple lineages. Specifically, we are interested in regions of the genome that are sufficiently linked that there is little evidence of recombination during pedigree meiosis. Further, there are two cases for these regions. First, if there is little evidence of ancestral recombinations or recurrent mutations in the founding haplotypes, the perfect phylogeny model [35] would apply to the pedigree haplotypes. The perfect phylogeny model has been shown to be realistic as long as the studied region is physically short [21, 22, 28]. Second, if there is evidence of ancestral recombinations in the region, then ancestral recombinations must be allowed, and the founding haplotypes are not restricted to a perfect phylogeny. We make no other assumptions about recombination rates or founder allele frequencies. These two cases allow us to make simplifying assumptions and allow efficient computation over large and complex pedigrees without compromising accuracy.

To solve the first case of the problem, we propose a blocked Gibbs sampler with running time polynomial in the number of SNPs and linear in the number of individuals. Roughly, PhyloPed, our method, chooses overlapping blocks of individuals that correspond to lineages in the pedigree. A single sampling step updates the haplotype assignments for all the individuals in the lineage of interest. The algorithm considers each lineage in turn, updates that lineage, and continues until convergence. PhyloPed begins the blocked Gibbs sampler at an initial state that is a feasible haplotype configuration that is compatible with the perfect phylogeny. In practice, the initial haplotype state can often be obtained quickly, though in the worst case, due to disallowing recombination, the running time may be exponential in the number of individuals. In the case that the founder haplotypes could not have come from a perfect phylogeny, PhyloPed reverts to the second case, without the perfect phylogeny, and runs the same blocked Gibbs sampler from an initial haplotype configuration with unrestricted founder haplotypes (with running time exponential in the number of SNPs). Furthermore, PhyloPed does not require knowledge of recombination rates or founder-allele frequencies. The perfect phylogeny allows more accurate haplotype inference, for a small number of SNPs, and yields inferences that can substantially improve the power to detect disease linkage.

4.1.1 Methods

According to convention, assume that every non-founder has both their parents represented in the pedigree. For each of the M bi-allelic SNPs, every individual w has an unordered single-locus genotype g_w^m at SNP m. An individual with a fully observed genotype has a single set of possible alleles $g_w^m \in \{\{0,0\},\{0,1\},\{1,1\}\}$. An individual with an unobserved or partially observed genotype has several possible sets of alleles $g_w^m \in \{\{\{0,0\},\{0,1\},\{1,1\}\},\ \{\{0,0\},\{0,1\}\},\ \{\{0,1\},\{1,1\}\}\}$.

We denote a haplotype by a sequence of binary alleles, $h \in \{0,1\}^M$, where M is the number of SNPs in a region of the genome. Let the m-th allele of haplotype h be denoted by $h(m)$. A pedigree *state* associates an ordered pair of haplotypes (or multi-locus genotype), $s_w = (h_w^0, h_w^1)$, with each individual $w \in I$. The haplotypes of an individual are *consistent* with the observed genotypes provided that at each SNP m, all known alleles are represented in the haplotypes, meaning that $\{h_w^0(m), h_w^1(m)\} \subset g_w^m$. Let $C(w, s_w)$ be an indicator variable that is 1 when the haplotype state s_w is consistent with the observed genotypes. Let $S(w)$ be the set of all haplotype states that are consistent with w's observed genotype.

We assume that the M SNPs are tightly linked and are effectively unable to recombine when passed from one generation to the next. In other words, haplotypes are passed according to Mendelian inheritance. More precisely, we define an individual's haplotype state s_w

as *non-recombinant* when the haplotype h_w^0 inherited from the father, $p(w)$, exactly matches one of the father's haplotypes, $h_w^0 = h_{p(w)}^0$ or $h_w^0 = h_{p(w)}^1$, and similarly h_w^1, inherited from the mother, $m(w)$, matches one of the mother's haplotypes.

Mendelian inheritance gives the probability that each of the father's (or mother's) haplotypes are inherited by the child: $Pr[h_w^0 = h_{p(w)}^0 | h_{p(w)}^0 \neq h_{p(w)}^1] = 1/2$ and $Pr[h_w^0 = h_{p(w)}^0 | h_{p(w)}^0 = h_{p(w)}^1] = 1$. This assumption determines a family of probability distributions over states of the pedigree, given genotype data for some of the individuals. Our goal is to find the haplotypes that maximize the conditional distribution of pedigree states given genotype data for some individuals.

Lineage Decomposition. Rather than computing the joint distribution of haplotype assignments to all the founders, we decompose a complex pedigree into tree-like lineages. Roughly speaking, each lineage is a block of variables that will be updated in a single iteration of the blocked Gibbs sampler, and the lineages are not necessarily disjoint from each other.

A lineage is defined as follows. Let $H(w)$ denote the set of children of node w. Founders p and q are called a *monogamous founding pair (MFP)* if and only if $H(p) = H(q)$ (i.e., there are no half-siblings of the founders' children). Assume that the pedigree contains only monogamous married pairs of founders. The *lineage of the monogamous founding pair* (p, q) is the induced subgraph of the pedigree that contains all the descendants of p and q. Formally, the lineage $L(p, q)$ is a directed, acyclic graph that contains a source node for each p and q and a node for each descendant of (p, q). This means that $L(p, q)$ is the smallest subgraph of the given pedigree such that L contains both founders p and q and, if L contains a node w, then L contains the children of w. If a parent of w is not in L then that parent is called the *non-lineage* parent.

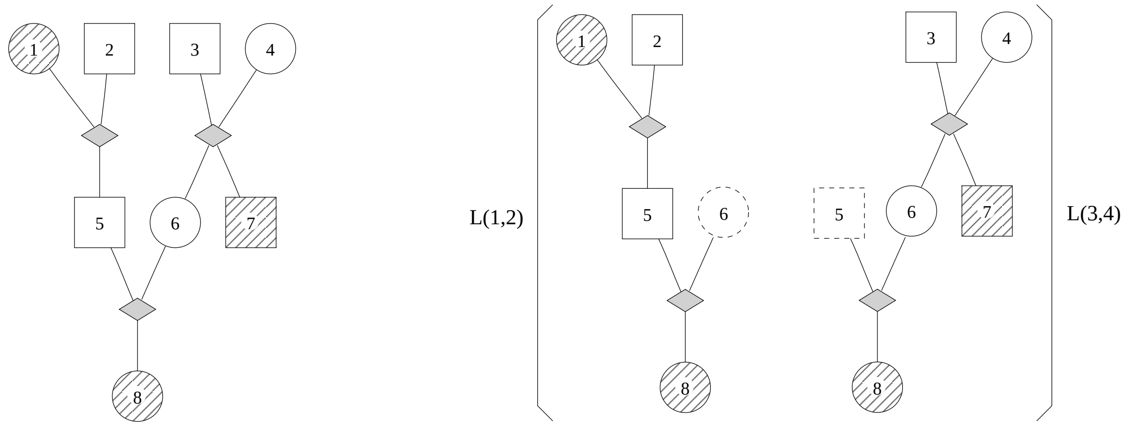


Figure 4.1: **Lineage Decomposition** The left panel shows the whole pedigree while the right panel shows the lineages of the pedigree. The non-lineage parents are dashed, and individuals 6 and 5, respectively, are parents of individuals in the lineages $L(1, 2)$ and $L(3, 4)$

For example a pedigree and its lineages are shown in Fig. 4.1 as two distinct pedigree sub-graphs. Notice that parents of lineage members fall into four categories: 1) founders participating in the MFP, and 2) non-lineage founding parents, 3) non-lineage parents (non-founders who are descendants of another lineage) and 4) lineage descendants (descended from the MFP).

Our goal is to chose a haplotype state for the pedigree from the posterior distribution of haplotype states given the genotype data of the pedigree and assumptions about haplotype sharing between lineages. A side effect of this is that PhyloPed infers all missing alleles, including haplotypes (and genotypes) for ungenotyped individuals. To find a haplotype state, we consider each lineage separately and calculate the distribution of haplotype states for the monogamous pairs of founders, given the genotype data of their descendants and probabilistic assumptions about the states of the non-lineage founders and non-lineage parents. The calculation proceeds in three phases:

1. Find a consistent, non-recombinant state for the pedigree. If there is such a state that is also compatible with some perfect phylogeny on all of the observed genotypes [35, 28], choose that state. Otherwise, when the founder haplotypes require ancestral recombinations or back mutations, choose any consistent, non-recombinant state for the pedigree haplotypes.

2. Decompose the pedigree into lineages, as described above.

3. Iterate over the collection of lineages: first, compute the distribution for the MFP haplotypes conditioning on the genotypes in the lineage and conditioning on the current states of the non-lineage parents, and second, sample new haplotype states for the lineage descendants from the computed distribution.

Branch-and-Bound for Non-recombinant Haplotype Inference

This algorithm initializes the algorithm in the previous section. Here we describe how to obtain a consistent, non-recombinant pedigree state (with a pair of haplotypes for every individual). We use a branch-and-bound algorithm that branches on possible haplotype states for each nuclear family and bounds consideration of haplotypes states when those haplotypes would yield a lower transmission likelihood than a previously observed pedigree state. The transmission likelihood for a pedigree state $\{s_w|w \in I\}$ is defined as:

$$\prod_{w \in I \setminus F} C(w, s_w) Pr[s_w | s_{p(w)}, s_{m(w)}].$$

Preprocessing. The preprocessing step is a linear-time operation that reduces the number of branching options that our algorithm will take. Assume the pedigree is ordered topologically. This means that the first node considered will be an individual without children and that children must be considered before their parents. Each nuclear family is considered in topological order until reaching the founders. Each trio is quickly considered to eliminate haplotype pairs for each individual that are inconsistent with the genotypes of other trio members. Recall that $S(w)$ is defined as set of haplotype pairs that are consistent with the observed genotypes of individual w. Typically $|S(w)| = 2^{N_w}$ where N_w is the number of heterozygous loci in w's genotype. However during the preprocessing step means that we effectively reduce the size of $S(w)$ by removing haplotype pairs that are inconsistent with the genotypes of w's nuclear family.

Alternatively, we can choose to restrict our attention to the subsets $P(w) \subset S(w)$ having the property that $H = \{P(w)|w \in I\}$ are haplotypes compatible with a single perfect phylogeny tree. For haplotypes this is equivalent to the stipulation that the evolution of the founding haplotypes occurred without recombination according to the infinite-sites coalescent model [35]. We use the *BuildTree* algorithm in [28] to quickly find the $o(2^{M-1})$ perfect phylogeny trees compatible with the genotyped individuals, where M is the number of SNPs.

Branching. For each nuclear family (in topological order), the algorithm branches on each possible haplotype assignment to the two parents. We get this list of assignments by listing the possible assignments to both parents that are consistent with the genotypes of the children. Once a haplotype assignment is made to both the parents, the children are assigned haplotypes (if they were not already assigned when considering a previous nuclear family). The haplotypes for the children are chosen to maximize the transmission likelihood conditional on the assignments for the parents.

In order to get the consistent, non-recombinant haplotype-pairs for the parents, we use Boolean logic. For the first parent, we make a Boolean expression that yields all pairs of haplotypes that are consistent with the genotypes of all the children and one of the parents (irrespective of the other parent's genotype). This list contains only haplotype pairs that maintain Mendelian consistency with the children. Once we have the list of feasible haplotype assignments for this parent, we branch the computation and make a different assignment to the first parent in each branch.

The second parent's haplotypes are assigned conditional on the assignment already made to the first parent, and there is a similar Boolean expression that gives a list of haplotype-pairs that are now consistent with all the individuals in the nuclear family. Again, the algorithm branches once for each distinct haplotype assignment for the second parent.

Bounding. The branching process proceeds up the pedigree either until all individuals are assigned haplotypes, or until a bounding opportunity appears. For each individual encountered while proceeding up the pedigree, the algorithm updates the partial log of the transmission likelihood which contains terms for individuals appearing later in the topological ordering. If the partial log-likelihood becomes smaller than the log-likelihood for a previously observed state, then the algorithm ceases to follow that branch of the computation.

Inference Scheme

We will briefly introduce inference on a single lineage, then discuss multiple lineages. Finally, we will give the detailed equations for inference.

Inference for a Single Lineage. To describe step 3 in detail, we need to establish a few assumptions. First, assume that we have a consistent, non-recombinant state for the pedigree (for details, see below). Also, assume for the moment that there is a known prior probability for founder haplotypes, $\alpha(h)$ for the 2^m possible haplotypes. Each time inference is performed on a lineage, the algorithm removes inbreeding loops by randomly choosing an individual to condition on. This is done by successively finding the oldest inbred descendant (whose parents are not inbred) and flipping a coin to choose which parent will be designated the non-lineage parent for the duration of the iteration.

For a single, non-inbred lineage, we can compute the probability of the MFP haplotypes by conditioning on 1) the haplotype assignments of the non-lineage parents, 2) the genotypes, and 3) the prior probability α. The child of a non-lineage parent inherits either one of the two equally-likely non-lineage haplotypes, if the non-lineage parent has a haplotype assignment, or one of the 2^m possible founder haplotypes drawn from α, if the non-lineage parent is an ungenotyped founder. The prior and transmission probabilities yield a tree-like graphical model from which to learn the MFP haplotype distribution. For the lineage $L(p,q)$, let $\mathbb{P}_{p,q}(i,j,k,l)$ be the marginal probability of the haplotype assignment (i,j) to p and (k,l) to q conditioned on the genotypes in the lineage and the haplotype assignments of the non-lineage parents. This is a marginal probability, because it is computed by summing over possible haplotype assignments for lineage descendants.

Some fairly standard bottom-up dynamic-programming equations yield the MFP marginal $\mathbb{P}_{p,q}(i,j,k,l)$ and incomplete marginals, or messages, for the lineage descendants (see the peeling algorithm in [61]). The descendant marginals are incomplete, because they are computed by summing only over possible haplotype assignments to their descendants (rather than summing also over possible assignments to their ancestors), and are conditioned only on the genotypes of their descendants (rather than all of the lineage genotypes). For an MFP child, r, with two lineage haplotypes (i,j), define $\mathbb{P}_r(i,j)$ as the incomplete marginal probability of r having haplotypes (i,j) conditioned on the genotypes of r's descendants. Similarly, for all other lineage descendants, with one lineage haplotype, define $\mathbb{P}_w(i)$ as the incomplete marginal probability of w having lineage haplotype i conditioned on the genotypes of w's descendants (see below for equations).

When considering all possible haplotype assignments, computation of these conditional probabilities takes time $O(N^4 \cdot L)$ where $N = 2^m$ is the number of possible haplotypes and L is the number of individuals in the lineage. Recall that m is small and the computation is feasible, because all the SNPs are in a short region of the genome and are in linkage disequilibrium. In cases where only perfect phylogeny haplotypes are considered, the running time is reduced to $O((m+1)^4 L)$. This follows from the fact that a perfect phylogeny contains at most $m+1$ haplotypes.

In order to update the haplotype state of a lineage, we use a top-down random propagation algorithm that chooses a new pair of haplotypes for each individual in the lineage (similar to the random propagation algorithm described in [61]). Random propagation allows us to choose haplotype assignments for each person from the correct, or complete, marginal haplotype distribution for that individual. For the pair of founders, p,q, haplotypes (i,j,k,l) are chosen proportional to $\alpha(i)\alpha(j)\alpha(k)\alpha(l)\mathbb{P}_{p,q}(i,j,k,l)$. Children, r, of the

MFP are randomly assigned haplotypes $(h_r^0, h_r^1) \in \{(i,k),(i,l),(j,k),(j,l)\}$, conditional on the MFP haplotype assignment (i,j,k,l), with probability proportional to $\mathbb{P}_r(h_r^0, h_r^1)$. All other lineage descendants, w, are given a lineage haplotype h_w conditional on the haplotypes $(h_{l(w)}^0, h_{l(w)}^1)$ of their lineage parent $l(w)$. So, $Pr[h_w^0 = h_{l(w)}^0]$ is proportional to $\mathbb{P}_w(h_{l(w)}^0)$. And the non-lineage haplotype h_w^1 is chosen from the set $\{h_{n(w)}^0, h_{n(w)}^1\}$ of haplotypes for the non-lineage parent $n(w)$ and probability proportional to $\alpha(h_w^1)C(w, h_w^0, h_w^1)$. The random propagation scheme is also accomplished in time $O(N^4L)$, except when perfect phylogeny haplotypes are known, making the running time $O((m+1)^4L)$.

Inference for Multiple Lineages. We now extend our algorithm to consider several monogamous founding pairs simultaneously. We no longer assume that there is a fixed α distribution or that the haplotype states never change. Instead, we use an iterative process that computes a new haplotype distribution α^t at each iteration t and maintains a consistent, non-recombinant haplotype state for all the individuals in the pedigree. For each iteration, t, consider each MFP (p,q) and its lineage $L(p,q)$:

1. Given the previous estimate of α^{t-1}, perform the bottom-up dynamic programming calculation to compute $\mathbb{P}_{p,q}^t(i,j,k,l)$, $\mathbb{P}_r^t(i,j)$ and $\mathbb{P}_w^t(i)$ for the MFP (p,q).
2. Use α^{t-1} together with the various $\mathbb{P}^t$ probabilities in the random propagation scheme to sample a new haplotype state for the individuals in the lineage.

After obtaining an updated $\mathbb{P}_{p,q}^t(i,j,k,l)$ for each MFP (p,q), compute the updated prior distribution as the marginal average $\alpha^t(h) \propto \sum_{(p,q)} m_{p,q}^t(i) + m_{p,q}^t(j) + m_{p,q}^t(k) + m_{p,q}^t(l)$ where the marginal $m_{p,q}^t(i) = \sum_j \sum_k \sum_l \mathbb{P}_{p,q}^t(i,j,k,l)$ and similar definitions apply for $m_{p,q}^t(j)$, $m_{p,q}^t(k)$, and $m_{p,q}^t(l)$. The iterations continue until the l_1 deviation between α^t and α^{t-1} falls below a pre-determined threshold. Clearly the running time of our method depends on the number of iterations until convergence. In practice, the l_1 deviation of the α^t estimates drop rapidly and most of the blocks in Fig. 4.2 converged in roughly 6-8 iterations (data not shown).

Correctness. We have described a blocked Gibbs sampling scheme where in each iteration, the updated block is a non-inbred subgraph of a pedigree lineage. Each update step uses a mixture of bottom-up recursion and top-down sampling to update the haplotype assignments in each block. A Markov Chain employing this update algorithm will converge to the correct posterior probability distribution when the haplotype states of the pedigree form an irreducible state space. In each update iteration, the haplotypes for the lineage individuals are updated conditional on the haplotypes assigned to the non-lineage parents, while the haplotypes of the non-lineage parents and all other pedigree individuals are unchanged. If the unchanged haplotypes are drawn from the stationary distribution, then after an update, all the haplotypes together represent a sample from the stationary distribution. This would by true for any blocking scheme, but we have chosen the lineage blocking scheme for ease of computation.

Details for Updating Lineage Haplotype Assignments

We describe how the haplotype assignments in a single lineage are updated within one iteration of our algorithm. Assume that we have a consistent, non-recombinant state for the pedigree (obtained by the branch-and-bound method). For the moment, consider restricted instances of a general lineage $L(p, q)$ for founders p and q. Further, consider the subgraph of $L(p, q)$ that contains only one child, r, of MFP (p, q). Call this subgraph L_r. For these restricted instances we assume:

1. L_r is a tree; *i.e.*, there is no inbreeding. This means that except for r, who has two lineage haplotypes, each node in L_r has one non-lineage parent $n(r)$ and one lineage parent $l(r)$ in L_r. If a lineage individual has two non-inbred parents in the lineage, flip a coin to choose which parent will be the non-lineage parent for the current update step.

2. We assume that each non-lineage founder is the parent of at most one node in L_r.

3. For each non-lineage parent v, the haplotype that v transmits to its child in L_r is drawn from a given prior probability distribution α, and the haplotype transmissions from the different non-lineage parents are mutually independent. For a non-lineage founder, all haplotypes are considered transmittable. However, for non-founder, non-lineage parents, the set of transmittable haplotypes is limited to $\{h^0_{n(r)}, h^1_{n(r)}\}$ and must be from a consistent, non-recombinant state of the pedigree

Our algorithm is similar to the peeling plus random propagation scheme described in [61] and has two steps, the Elston-Stewart dynamic programing step moving up the lineage (see Section 3.2.1 for details) and the random propagation of new haplotype assignments down the lineage.

Random Propagation. Once we have computed the quantities $\mathbb{P}_{p,q}(i, j, k, l)$, $\mathbb{P}_r(i, j)$ and $\mathbb{P}_w(i)$ for the MFP (p, q) and their descendants we can sample efficiently from the conditional distribution of the joint assignments of haplotype pairs to all nodes of $L(p, q)$, subject to the requirement that the haplotype assignment at each node is compatible with a consistent, non-recombinant state for all their descendants. Each sample is computed by traversing $L(p, q)$ top-down, starting at the source nodes p and q. The haplotype pair assigned to each node is determined after the haplotype pairs assigned to its parents have been determined.

In detail, the process is as follows. As before, let i, j denote the two haplotypes of p and let k, l the two haplotypes of q. The probability, in the sample, that the haplotypes of p and q are set to the particular values (i, j, k, l) is proportional to $\alpha(i)\alpha(j)\alpha(k)\alpha(l)\mathbb{P}_{p,q}(i, j, k, l)$, where the constant of proportionality makes the probabilities of all choices sum to 1. Similarly, let r be a child of the founding pair. Then the probability that r's haplotypes are set to (i, k) is proportional to $\mathbb{P}_r(i, k)$, and a similar formula holds for the three other possible choices (i, l), (j, k), and (j, l). If w is a node of $L(p, q)$ whose parent z is not p,q, or r then the sampling process sets the lineage haplotype as either h^0_z or h^1_z with probability proportional to $\mathbb{P}_w(h^0_z)$ and $\mathbb{P}_w(h^1_z)$. The probabilities of the other joint choices are computed similarly.

The execution time of the sampling process is $O(N^4L)$, where L is the number of individuals in the lineage, and $N = 2^m$ is the number of haplotypes on m loci.

We now extend our algorithm to deal with violations of the tree-like model. The violations are of two types:

1. A non-lineage founder may be a parent of more than one node among the descendants of the monogamous founding pairs.

2. A node may be the child of two nodes descended from the same MFP. In that case, one of the parents is arbitrarily designated as the non-lineage parent and the edges from such non-lineage parents are excluded in forming the dags $L(p, q)$ descending from the monogamous founding pairs. As a result, each descendant of a monogamous founding pair lies in exactly one of these dags.

4.1.2 Results

Pedigree Simulations. In order to test the accuracy of our method, we simulated a set of pedigrees with their corresponding haplotypes. Given a pedigree, founder haplotypes were generated uniformly at random from the phased HapMap CEU haplotypes for Chromosome 1. We considered only common SNPs (with minor allele frequencies at least 0.05). We performed multiple trials, where each trial consisted of a distinct sample of SNPs chosen to have a specific density along the genome. This allowed us to vary the mean physical distance between neighboring SNPs. Each sample of SNPs was arbitrarily partitioned into non-overlapping blocks of a fixed length for haplotype inference.

The non-founders were generated in successive generations using Poisson-distributed recombinations (without interference), where the recombination rate was a function of the physical distance, such that there is an average of two recombinations on the length of Chromosome 1. Considering each non-founder in turn, we obtained one haplotype from each parent by uniformly choosing one of the parental haplotypes to provide the allele for the first SNP. Alleles for successive SNPs were chosen either to be non-recombinant or recombinant according to the recombination rate. We refer to the complete simulation output (of phased haplotypes) as the *gold-standard data*.

We chose pedigrees with fixed structure. For each pedigree we fixed the number and set of individuals to be genotyped in the data input to each of the phasing algorithms (and removed the phase information for all of the ungenotyped individuals).

L1 10 copies of a 20-individual family with 1 lineage and exactly 13 genotyped individuals (1000 blocks of 3 SNPs with 11kbp between SNPs)

S1 single family with 10 lineages and 59 individuals, exactly 24 of them being genotyped on 1000 blocks of 3 SNPs.

M1 5 copies of the family from S1, with exactly 24 individuals genotyped in each family (1000 blocks with 3 SNPs).

M2 10 copies of a 10-individual family with 2 lineages and exactly 5 genotyped individuals (10,000 blocks of 3 SNPs).

H1 single 16-individual, 2-lineage pedigree with half siblings and exactly 9 genotyped individuals on 300 blocks of 5 SNPs.

R1 60 copies of a 12-person family with 2 lineages and 12 individuals with 6 genotyped individuals on blocks of 4 SNPs.

R2 30 copies of a 7-person nuclear family with 5 full siblings and 5 children genotyped on blocks of 6 SNPs

Haplotype Inference Accuracy.

We compared our approach to two others, Merlin [16] and Superlink [29]. Merlin and Superlink perform a maximum-likelihood calculation on a similar graphical model of inheritance in a pedigree, where recombination rates and founder allele frequencies are given as fixed parameters of the model. However, Merlin employs a different elimination order for the EM algorithm than does Superlink and has an option for non-recombinant haplotype inference (this option was not used here, because it seemed to make little difference to inference accuracy). PhyloPed uses a graphical model of inheritance that is similar to that used by Merlin and Superlink but does not require the founder allele frequencies or recombination rates.

The input data consisted of the pedigree relationships and the genotype data for only the typed pedigree members. Merlin and Superlink were additionally provided with the correct recombination rates and with either an uninformative prior for the founder alleles or the perfect prior (i.e., the correct allele frequencies). Every phasing program was run on consecutive, non-overlapping blocks of k SNPs, and all programs ran on the same blocks. The output of each of the phasing programs was compared to the gold-standard data, and again the comparison used the same k-sized blocks. In cases where phasing programs provided a list of possible phasings, the first phasing was tested for accuracy. Accuracy was measured as the percentage of haplotype assignments in the phase estimate that matched the haplotypes in the gold-standard haplotype data. Notice that in this definition of accuracy the parental origin of the haplotype is irrelevant. Notice also that if the assumptions of PhyloPed are not satisfied, meaning that a particular family required recombinant haplotypes, then PhyloPed produced no estimate, and we conservatively chose to penalize our method by scoring the lack of a prediction as zero accuracy.

Simple vs Complex Pedigrees. For the single-lineage pedigree L1, we simulated blocks of size $k = 3$ with the average physical distance between SNPs being 11kbp. All methods estimated haplotypes with similar accuracy (Table 4.1, row L1). This suggests that the models have few practical differences on simple pedigrees.

For multi-lineage pedigrees S1, M1, M2, and H1, we see that PhyloPed outperforms the other methods (Table 4.1, rows S1,M1, and H1, and Fig. 4.2). Most of these results were generated for blocks with $k = 3$ SNPs, because larger blocks were infeasible for Merlin.

		Perfect Prior		Uninformative Prior	
Pedigree	**Method**	**Avg**	**Std-Dev**	**Avg**	**Std-Dev**
L1	PhyloPed	**0.867**	0.030	**0.867**	0.030
	Merlin	0.855	0.018	0.857	0.018
	Superlink	0.836	0.034	0.819	0.023
S1	PhyloPed	**0.809**	0.065	**0.809**	0.065
	Superlink	0.796	0.064	0.642	0.066
M1	PhyloPed	**0.808**	0.060	**0.808**	0.060
	Superlink	0.795	0.058	0.636	0.058
H1	PhyloPed	**0.816**	0.161	**0.816**	0.161
	Merlin	0.750	0.138	0.761	0.124
	Superlink	0.799	0.116	0.717	0.148

Table 4.1: **Average accuracy and standard deviation.** In all cases, PhyloPed dramatically outperforms Merlin. PhyloPed is substantially better than Superlink, when given a non-informative perfect prior. When given the perfect prior, Superlink performs no better than PhyloPed. In cases where Superlink and PhyloPed have comparable performance, we see that the uninformative prior particularly hurts Superlink's accuracy. Merlin was unable to execute S1 and M1 due to running time.

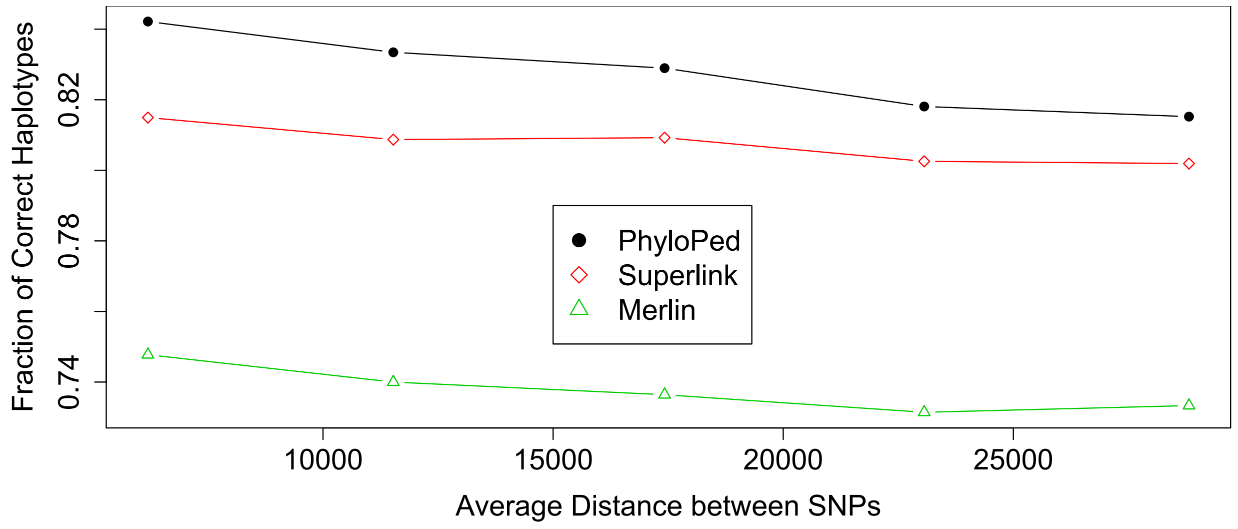


Figure 4.2: **Accuracy Against Recombination Rate.** This plot shows results of 10,000 blocks for M2 the 2-lineage, 10-individual family. The accuracy of each method was computed for different physical distance between neighboring SNPs.

However, pedigree H1 was simulated with $k = 5$ SNPs, and still PhyloPed outperforms the others.

We also see in Table 4.2 that each method is favored under different pedigree types. Our method performs roughly comparably to Merlin, while Superlink clearly has an edge in speed in all instances except the half-sibling case.

Pedigree	Method		Avg Run-Time
L1	PhyloPed		0.150 s
	Merlin		0.166 s
	Superlink		0.072 s
S1	PhyloPed		0.612 s
	Superlink		0.041 s
M1	PhyloPed		18.2 s
	Superlink		0.156 s
H1	PhyloPed		3.69 s
	Merlin		0.088 s
	Superlink		17.5 s

Table 4.2: **Average run-time per block.** Each program has different run-time strengths. PhyloPed compares favorably to both Merlin and Superlink on different types of pedigrees.

Violations of Assumptions. We consider two violations of the assumptions for the three methods. First, we consider the performance of the three methods for different physical distances between SNPs in the block (resulting in a range of recombination rates). PhyloPed consistently outperforms Superlink and Merlin even as the recombination rate increases (Fig. 4.2). Second, it is possible for the founder allele frequencies to be unknown, even while the recombination rates may be known. We provide both Superlink and Merlin with uninformative founder allele frequencies (i.e. frequency 0.5 for all alleles). In this scenario, Merlin performs comparable to when it is given a perfect prior, but Superlink's accuracy decreases dramatically.

Power to Detect Disease Linkage.

Here we examine the influence that haplotype inference may have on the ability to correctly detect disease linkage. We consider three disease detection methods: 1) MQLS [96], a quasi-likelihood association test, 2) Merlin's linkage test [1], and 3) Merlin's association test [14]. Specifically, we are interested in whether haplotype inference before disease detection, will improve the power to detect the disease without many more false positives. We consider haplotypes inferred by both Superlink and PhyloPed, with our method drawing multiple samples from the Gibbs sampler and retaining only individual haplotypes that appear in at least 80% of the samples.

We simulated 500 replicates of genotype data respectively for pedigrees R1 and R2 exactly as described above. Then, for each replicate, we select a SNP to be the disease locus and hide all genotype data for that SNP from the inference and testing programs. The two disease models of interest are:

Alternative: disease status being a random function of the disease locus via the penetrance probabilities that result in a population prevalence of 0.1 and relative risk 5;

Null: disease status being randomly chosen, such the total number of affected individuals is the same as in the alternative model.

In both these models, we ensure that at least 20% of the individuals in the pedigree have the disease. This results in genotype data at 3 SNPs and 5 SNPs respectively for the R1 and R2 pedigrees. Recall that ungenotyped individuals are simulated by hiding the genotypes of 6 and 2 individuals respectively.

We compare the various methods by plotting a receiver-operator curve (ROC) showing the true positive and false positive rates of each test for different test-score cutoffs. The main advantage of this plot is that it allows comparison without requiring an analytical null distribution for each test. A superior method has an ROC that appears above and to the left of those for other methods.

For both the R1 and R2 pedigrees, the most accurate disease detection method is to use the MQLS test on the PhyloPed inferred genotypes, see Figures 4.3 and 4.4. This is more accurate than simply using the MQLS test or the Merlin linkage test without any inference, presumably due to the missing genotypes. Notably, MQLS consistently gave the best power for linkage detection without inference of missing data. Using the linkage test after inferring genotypes with PhyloPed seems to result in little consistent advantage, probably due to the test integrating over all inheritance possibilities. That approach is advantageous when the estimate has very high confidence, such as with R2, but otherwise it performs similarly to the linkage test alone (data not shown). Furthermore, when inferring genotypes with Superlink on many different pedigrees, we observed a consistent disadvantage, unless there is little missing data (data not shown for Superlink+Merlin). The Merlin association test performs its own genotype inference prior to using a regression test for association, and surprisingly performs much better than the Merlin linkage test in Figure 4.3.

4.1.3 Summary

We have introduced PhyloPed which leverages the population genetics of the founders to produce superior haplotype estimates for multi-lineage pedigrees. Specifically, PhyloPed assumes that the founder haplotypes are drawn from a perfect phylogeny and that haplotypes are inherited without recombination in the pedigree. As we have shown, this approach works very well for short regions with dense SNPs. Not only does our method provide accurate haplotype inference, but also these inferences substantially improve power for disease detection when a sufficient amount of genotype data on linked SNPs is available. We recommend that pedigree association studies analyze their data with a combination of PhyloPed and MQLS, or MQLS alone if inference is impractical.

In addition to the perfect phylogeny model, there are several other reasons that PhyloPed outperforms other methods. Intuitively, Occam's razor suggests that our method would be preferable on blocks having little recombination. Assuming no recombination provides not only fewer phasing options to consider but also fewer parameters and less over-fitting. PhyloPed requires no prior information, neither for the recombination rates nor for the founder allele frequencies, making it more robust, even with inaccurate priors.

Many factors influence the accuracy of haplotype estimation, including the complexity of the pedigree, the number and relationships of genotyped individuals, and the number of

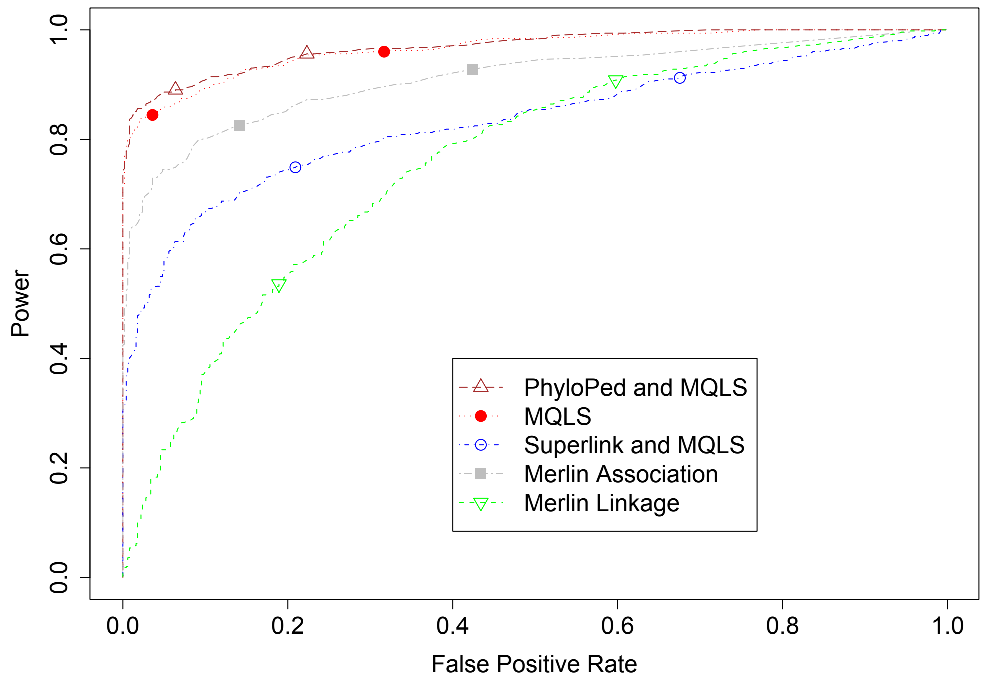


Figure 4.3: **ROC Plot for Tests With and Without Genotype Inference.** Pedigree R1 was tested at 3 SNPs for disease linkage after hiding a linked disease SNP. Of the 4 simulated SNPs, neighboring SNPs were separated by roughly 6kbp.

linked SNPs. The number of genotyped individuals in the pedigree and their relationships with the other pedigree members influence the number of constraints available for haplotype estimation. Typically, having genotypes for more individuals yields better haplotype estimates. Similarly, simultaneous phasing of larger numbers of linked SNPs can reveal more haplotype information, provided that the pedigree is not so large that the computational burden is infeasible. This section has focused on inference in deep and complex pedigrees and partitioned the genome into blocks before phasing. In order to properly treat the whole genome, future research should consider partitioning schemes and methods for producing whole genome haplotype estimates from the estimates for each partition.

Pedigrees should not be made unnecessarily complex. Multiple-lineage pedigrees are only useful in the case where each founding lineage provides information about either the phenotype or the relatedness of genotyped individuals. For example, estimation of haplotypes in a nuclear family whose members are genotyped and phenotyped would not benefit from the introduction of grandparents whose additional degrees of freedom provide no additional constraints on the haplotypes or phenotypes. However, if a pair of grandparents are the common ancestors of this nuclear family and another genotyped family, then the grandparents' presence in the pedigree (along with the additional family) would provide

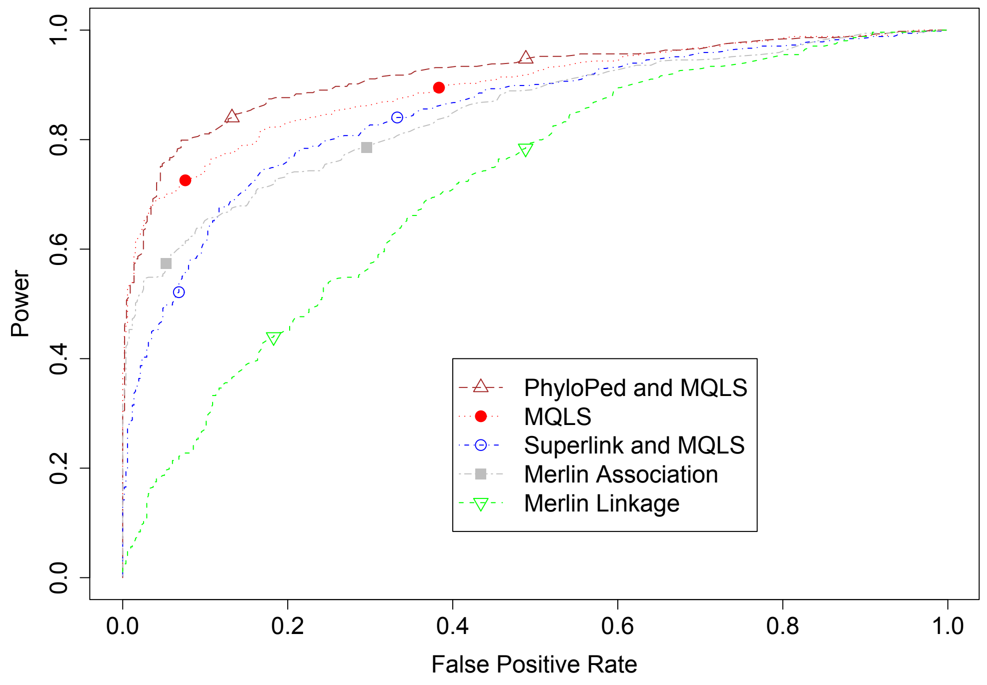


Figure 4.4: **ROC Plot for Tests With and Without Genotype Inference.** Pedigree R2 was tested at 5 SNPs for disease linkage after hiding a linked disease SNP. Of the 6 simulated SNPs, neighboring SNPs were separated by roughly 25kbp.

useful constraints.

4.2 Haplotype Hidden Markov Model

The work in this section was a novel contribution by the author [50, 48].

Single-molecule sequencing is an attractive alternative to genotyping and may soon yield haplotypes for individuals in a pedigree [26]. Sequencing methods would apparently yield more information from the same set of sampled individuals, which is critical due to the limited availability of individuals for sampling in multi-generational pedigrees (i.e. individuals usually must be living at the time of sampling). There is substantial evidence that haplotypes can be more useful than genotypes for both population and family based studies when using methods such as association studies [5, 16] and pedigree analysis [14, 51]. While it is intuitive that haplotypes provide more information than genotypes, there are instances with family data in which there are few enough typed individuals that there is little practical difference between haplotype and genotype data. Since we have already determined the complexity of likelihood calculations given haplotypes in Chapter 3, the goal of this section is to discuss algorithms for haplotype data and compare them to algorithms for genotype data.

To compute the analogous probability for haplotype data, we use a similar HMM to the one given in Section 3.2.2. For haplotypes, the hidden states must consider the *haplotype orientations*, which specify the parental origins of all the observed haplotypes. Notice that these orientations are not equivalent to inheritance paths, since they only specify inheritance edges between haplotyped individuals and their parents. For each of the $2^{2|H|}$ haplotype orientations, where H is the set of haplotyped individuals, we enumerate the inheritance paths compatible with the haplotype alleles, their orientations, and the pedigree relationships. Alternatively, each of the inheritance paths enumerated for the genotype algorithm induces a particular orientation on the haplotypes heterozygous for that locus (i.e. parental origin of the entire haplotype). Thus, the hidden states for the haplotype HMM are the cross-product of the orientations and the inheritance paths.

The haplotype HMM has transition probabilities that are nearly identical to the genotype HMM with the exception that transitions between inheritance paths with different haplotype orientations have probability zero. Recombinations are only allowed when they do not occur between typed haplotypes.

The forward-backward algorithm is also used on the haplotype HMM. However, there are $2^{2(|I|+|H|-|F|)}$ hidden states, yielding a slightly slower calculation. Fortunately, the haplotype recursions can be run simultaneous with the genotype recursions, meaning that the inheritance paths need only be enumerated once.

Haplotype Likelihoods in Linear Time

There is one obvious instance of the haplotype likelihood problem where there is a polynomial-time algorithm. Even though it is impractical to assume that we can sample genetic material from deceased individuals in a multi-generational pedigree, for a moment, let us consider the case where all the individuals in the pedigree are haplotyped.

The Elston-Stewart algorithm [27] for genotype data has a direct analogue for haplotype data. This algorithm calculates the likelihood via the belief propagation algorithm by eliminating individuals recursively from the bottom up. Each individual is “peeled off”, after their descendants have been peeled off, by using a forward-backward algorithm on the HMM for the mother-father-child trio.

The haplotype version of this algorithm is linear when all the individuals are haplotyped, since each elimination step is conditionally independent of all the others. Given the parents’ haplotypes, regardless of which was inherited from which grand-parent, the probability of the child’s haplotype is independent of all other trios. Therefore, we can take a product over the likelihoods for all the trios, and compute each trio likelihood using a 4-state HMM. Then for k non-founding individuals, and l loci, this algorithm has $O(kl)$ running time.

This same intuition carries through to the minimum recombination problem, and each trio can be considered independent of the others. This contrasts with the genotype minimum recombination problem which is known to be hard, even when all the individuals are genotyped [62].

Results

To simulate realistic pedigree data, SNPs were selected from HapMap that span 100mb on both sides of a loosely-linked pair of sites. There are 40 SNPs total, with 20 tightly linked SNPs on each side of a strong recombination breakpoint having $\theta = 0.25$. The haplotypes for these SNPs were selected randomly from HapMap. Pedigree haplotype and genotype data were simulated for each child by uniformly selecting one of the parental alleles for the first locus, and subsequent loci were selected on the same parental haplotype with probability θ_j for each locus j. Inheritance was simulated for 500 simulation replicates.

The simulation yielded completely typed pedigrees. For each pedigree, we removed the genotype and haplotype information for increasing numbers of untyped individuals. For each instance of a specific number of untyped individuals, two values were computed on the estimated number of recombinations between the central pair of loci: the haplotype and genotype accuracies. Accuracy was computed as a function of the l_1 distance between the deterministic number of recombinations and the calculated distribution. Specifically, accuracy was $2 - \sum_{i \geq 0} |x_i - a_i|$, where x_i was the estimated probability for i recombinations and a_i was the deterministic indicator of whether there were i recombinations in the data simulated on the pedigree.

In all the instances we observed a trend where the best accuracy was obtained with haplotype data where everyone in the pedigree was haplotyped. For example, a five-individual pedigree with two half-siblings is shown in Figure 4.5, left panel. With the three founders untyped, the haplotype data yielded similar accuracy as the genotype data. Consider a three-generation pedigree having two parents, their two children, an in-law, and a grandchild for a total of six individuals, three of them founders. This pedigree has a similar trend in accuracy as the number of untyped founders increases, Figure 4.5, right panel. As the number of untyped individuals increases, the accuracies of genotype and haplotype estimates appear to converge.

4.3 State-Space Reduction for HMMs

The work in this section was done as a collaboration between the author and Kirkpatrick [52].

Typical Hidden Markov Models (HMMs) on the genotypes of related individuals require an exponential number of hidden states. We look for maximal ensembles of the hidden states that can be used to create a new HMM with a more efficient running-time. We give a $O(n^2)$ algorithm that finds a group of isometries of the state-space which subsumes an existing algorithm by considering more isometry cases. From previous work, it is clear that finding the new state-space for an arbitrary group of isometries is NP-hard, and there is unlikely to be a polynomial algorithm for exploiting the state-space found by a polynomial-time isometry algorithm. So, we turn our attention to an exponential algorithm which provides optimal state-space reductions.

We settle an open problem by giving an example showing that a permutation group on the 2^n vertices of the hypercube can produce larger ensemble states than a group of isometries

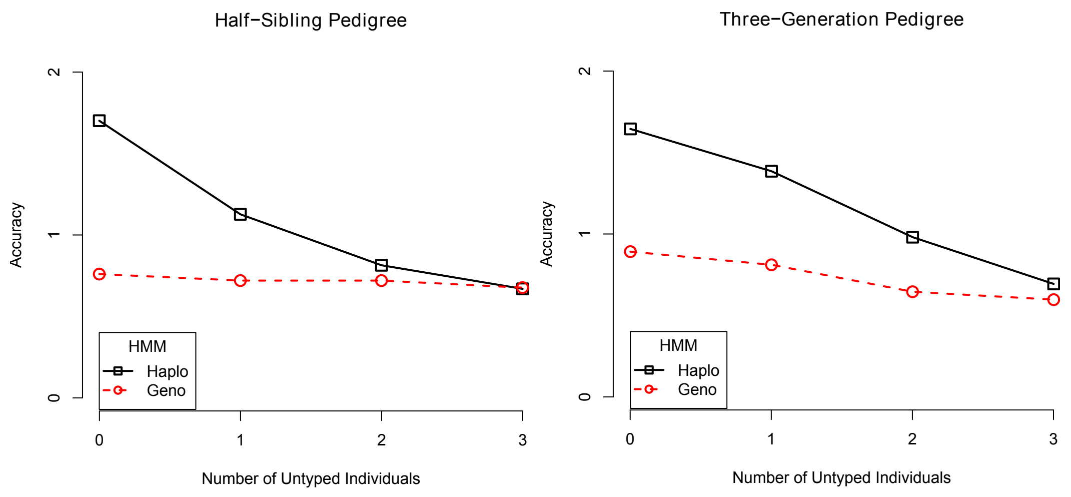


Figure 4.5: **Predicting Recombinations.** The left panel is the average accuracy for predictions from a pedigree with two half-siblings and three parents. The right panel shows results from a six-individual, three-generation pedigree. In both cases, 500 simulation replicates were performed, and the average accuracy of estimates from the haplotype data is superior to those from genotype data. However, as the number of untyped founders increases, in both cases, the accuracy of estimates from haplotype data drop relative to the accuracy from genotype data. The accuracies of genotype and haplotype estimates appear to converge.

on the n dimensions of the hypercube. We introduce an improved algorithm for finding the maximal ensemble states, sets of hidden states, that preserve both the Markov property and the identity by descent (IBD) information of the individuals of interest. Our algorithm has run-time $O(n4^n)$ as compared with previous algorithms having run-time $O(n!2^{2n})$, where n is the number of meioses in the pedigree.

4.3.1 Introduction

Motivation. Estimates of probabilities on pedigrees are of great interest to computer scientists because they give an important example of graphical models which model probability distributions by using a graph whose edges are conditional probability events and whose nodes are random variables [61]. Methods for reducing the state-space of a pedigree graphical model could generalize to other graphical models, as noted also by Geiger et al [32].

The Problem Summary. Hidden Markov Models (HMMs) analyzing the genotypes of related individuals typically take exponential–or even super-exponential–running time, so it is desirable to find more efficient algorithms. Any partitioning of the state space into ensemble states (i.e., states with identical emission probabilities and comparable transition probabilities) will improve the running time of an HMM, even if the ensembles are not optimal: since these HMMs have an exponential state space and a running time polynomial in the size of the state space, even an exponential algorithm for finding ensemble states can improve the running time of the HMM calculations.

Literature Review. Donnelly [24] introduces the idea of finding ensemble states for the IBD Markov modeland uses a manual method for finding the symmetries for several examples of two-person pedigrees. Browning and Browning [12] formalize the requirements for symmetries that can be used as ensemble states in a new HMM. They give the first algorithm for finding the maximal set of isometries that preserve the Markov property and the IBD information. Their algorithm, however, appears to have worst-case running time of at least of $O(n!4^n)$, where n is the number of meioses in the pedigree. They leave open the question of whether groups other than isometry groups could give useful state-space reductions [12].

McPeek [66] presents a more careful formulation of the identity states and a naive algorithm. Most recently Geiger et al [32] give a special-case state-space reduction involving only isometries that collapse simple lineages (multiple generations with a single child per generation and with the non-lineage parents being founders).

Related problems have been considered in the systems literature. Junttila considered partitions of a state space of strings and discussed the computational complexity of finding representatives of each orbit where the representative is the lexicographically minimal string [43]. Several people have introduced algorithms for finding symmetries for systems applications [64, 44]

Our Approach. Inspired by the work of Browning and Browning [12], we look for maximal ensembles of the hidden states that can be used to create a new HMM with a much more efficient running-time. We introduce an improved algorithm for finding the maximal ensemble states, sets of hidden states, that preserve both the Markov property and the identity by descent (IBD) information of the individuals of interest. Not only does our algorithm run faster than previous approaches, but it also finds larger ensemble states.

First, we discuss the problem of finding a group of isometries, permutations, of the n dimensions of the hypercube. These isometries may not be the maximal group, i.e. the group with the largest number of elements. Browning and Browning introduced an algorithm that finds a maximal group of isometries, however their algorithm is super-exponential.

Then we introduce an algorithm for finding a permutation group on the 2^n vertices of the hypercube, and produces the most efficient ensemble states (i.e. the smallest partition of the state-space that respects the IBD and Markov properties and has the minimal number of sets in the partition). Notice that this algorithm optimizes a different objective function than that introduced by Browning and Browning, and indeed, their algorithm may not find the optimal partition of the state space, as we demonstrate by giving an example.

4.3.2 Problem Description

Consider a pedigree graph, P, having individuals V as nodes and having n meioses with each meiosis being a directed edge from parent to child. Let I being the set of individuals of interest (perhaps because we have data for those individuals). An *inheritance state or vector* is a binary vector x with n bits where each bit indicates which grand-parental allele, paternal or maternal, was copied for that meiosis. The equivalent *inheritance graph*, R_x, has two nodes per individual (one for each allele) and edges from inherited parental alleles to their corresponding child alleles. Individuals of interest are called *identical by descent (IBD)* if a particular founder allele was copied to each of the individuals. In general, the inheritance graph is a collection of trees, since each allele is copied from a single parent.

The set of all inheritance states (binary n-vectors) is the n-dimensional hypercube $\mathcal{H}_n$, with 2^n vertices. The inheritance process is modeled as a symmetric random walk on $\mathcal{H}_n$, with the time dimension of the walk being the distance along the genome. At equilibrium, the walk has uniform probability of being at any of the hypercube vertices. From vertex x in $\mathcal{H}_n$, a step is taken to a neighboring vertex after an exponential waiting time with parameter $\lambda = n$. For each individual zygote, with one meiosis, this is a Poisson process with parameter $\lambda = 1$ and genome length roughly 30.

There is a discrete version of this random walk, which is often used for hidden Markov models (HMMs) that compute the probability of observing the given data by taking an expectation over the possible random walks on the hypercube. Let X be a Markov process, $\{X_t : t = 1, 2, ..., m\}$ for m loci with a state space $\mathcal{H}_n$ consisting of all the inheritance states of the pedigree. The recombination rate, θ_t, is the probability of recombination per meiosis, between a neighboring pair of loci, t and $t+1$. If t and $t+1$ are separated by distance d, then the Poisson process tells us that the probability of an odd number of recombinations is $\theta_t = 1/2(1 - e^{-2\lambda d})$. The natural distance on $\mathcal{H}_n$ is the Hamming distance, $|x - y|$, for two states x and y. Then the probability of transitioning from x to y is

$$Pr[X_{t+1} = y | X_t = x] = \theta_t^{|x-y|}(1-\theta_t)^{n-|x-y|}.$$

Conventional algorithms for computing likelihoods of data have an exponential running time, because the state space of the HMM is exponential in the number of meioses in the pedigree. We propose new ways to collapse hypercube vertices into ensemble states for a new HMM that has a more efficient running time. In particular we are interested in optimal ensemble states that preserve certain relationship structures and the Markovianness of the random walk.

The relationship structures we wish to preserve are the IBD relationships on the individuals of interest I. Let I_m be the maternal alleles of all the individuals and I_p be the paternal alleles of all the individuals. We say that a partition J of all the alleles $I_m \cup I_p$ is *IBD* for the inheritance state x if for each element of J, every allele in that element appears in the same connected component of the inheritance graph R_x. If we write $CC(R_x)$ for the partition induced by the connected components of R_x, then J is IBD for inheritance state x if and only if $J = CC(R_x)$. The partition J induces an *IBD class* called D_J on the inheritance states:

$$D_J = \{x \in \mathcal{H}_n | J = CC(R_x)\}.$$

Let D_{e_i} be the identity IBD class, where each e_i is a set with a single element; this identity IBD class corresponds to the partition created by the identity permutation.

The condensed identity states introduced by Jacquard [42] are the IBD classes for a pair of inbred individuals. McPeek [66] mentions that there is a natural extension of the pairwise identity states to more generalized IBD classes for inbred pedigrees and proved that those IBD classes are equivalence classes on the emissions probabilities. Geiger, et al [32] simply stated the requirement that the ensemble states for a reduced HMM must induce an equivalence class on the emissions probabilities.

We define potential ensembles of states as being the orbits of a group of symmetries, where the symmetries map elements of an IBD class to other elements of the same IBD class. Let G be a group on the state space $\mathcal{H}_n$ of X. A symmetry is a bijection $\psi \in G$ where ψ is a permutation on 2^n elements, the vertices of $\mathcal{H}_n$. Furthermore, to preserve the IBD classes, all the permutations $\psi \in G$ must satisfy $\psi(x) \in D_J$ for all $x \in D_J$ and for all J. The *orbits* of G are sets

$$w(y) = \{x | x = \psi(y) \text{ and } \psi \in G\}$$

for all $y \in \mathcal{H}_n$.

A special type of symmetry group is a group of isometries, G_{iso}, where each isometry is a bijection $T \in G_{iso}$ on $\mathcal{H}_n$ where $|T(x) - T(y)| = |x - y|$ for all x, y. If G_{iso} is an isometry group that preserves the IBD classes, then the orbits of G_i can be the states of a new Markov random walk. Let Y be the new Markov process, $\{Y_t : t = 1, 2, ..., m\}$ on the same m loci having states $\{W_1, W_2, ..., W_k\}$, where each state is an orbit $W_i = w(y)$ for some y and $W_i \cup W_j = \emptyset$. Notice that the orbits define a partition on the vertices of the hypercube. The Markov process Y has transition rates

$$Pr[Y_{t+1} = W_j | Y_t = W_i] = \sum_{y \in W_j} Pr[X_{t+1} = y | X_t = x] \text{ for } x \in W_i. \tag{4.1}$$

Browning and Browning [12] demonstrated that a group of isometries will always produce ensemble states that are Markovian. They also showed that any isometry $T : \mathcal{H}_n \rightarrow \mathcal{H}_n$ can be uniquely written as $T = \pi \circ \phi_a$ where π is a permutation on n elements, the bits of the hypercube vertex, and ϕ_a is a switch function where $\phi_a(x) = a + x$ where $+$ is the bit-wise XOR operation.

Markov Property

Let Y be a new Markov processes, $\{Y_t : t = 1, 2, ..., m\}$ on the same m loci, having states $\{W_1, W_2, ..., W_k\}$, which are orbits of G. This new Markov chain is coupled to the original such that when $X_t = x \in W_i$, $Y_t = W_i$, and Y_t is a projection of X_t into a smaller state space. The transition probabilities are:

$$\sum_{y \in W_j} \sum_{x \in W_i} Pr[X_{t+1} = y | X_t = x].$$

Process Y, the expectation process with transition probabilities:

$$\begin{aligned} \frac{1}{|W_i|} Pr[Y_{t+1} = W_j | Y_t = W_i] &= \sum_{y \in W_j} Pr[X_{t+1} = y | X_t = x] \\ &= \mathbb{E}[E_j | X_t = x], \end{aligned}$$

where E_j is the event that $X_{t+1} \in W_j$. The equilibrium distribution is: $Pr[Y_t = W_i] = |W_i|/2^n$. In order for this to hold, we need to choose a group G having orbits such that

$$\sum_{y \in W_j} Pr[X_{t+1} = y | X_t = x_1] = \sum_{y \in W_j} Pr[X_{t+1} = y | X_t = x_2] \tag{4.2}$$

for all $x_1, x_2 \in W_i$ for all i. While isometries certainly satisfy this property, there are other groups that do also.

We also need to show that an isometry group G of has orbits such that

$$Pr[X_{t+1} = y | X_t = x] = Pr[X_{t+1} = T(y) | X_t = T(x)] \tag{4.3}$$

for all $T \in G$, $y \in W_j$ and $x \in W_i$ for all i, j.

Isometry Algorithm

Several authors have found groups of isometries that lead to state-space reductions. This approach was pioneered simultaneously by Browning and Browning [12] and by McPeek [66]. Browning and Browning introduced an exponential algorithm for finding the maximal group of isometries (equivalently the minimal partition of the inheritance vectors obtained by any group of isometries). More recently, Geiger, et al [32] made use of a particular type of isometry group that is detectable in $O(n^2)$ time.

Obtaining the State Space from Isometries. Once we have obtained a group of isometries, for instance by using the Geiger, et al. method, we need to obtain the orbits of this group of isometies. Notice that this might be accomplished by finding a representative element of each orbit and testing whether other elements belong to the same orbit or not [64, 44]. Since isometries include permutation groups, this problem is clearly also hard for groups of isometries. Additionally, we note that computing Equation 4.1 appears to require enumeration of the orbit W_j. No matter how we obtain a group acting on the inheritance vectors, whether it is an isometry of the axis of the hypercube or a permutation on the vertices of the hypercube, enumeration of the orbits of the group appears to be required.

Given an arbitrary group of isometries, we give an exponential algorithm for enumerating the orbits of the group. This algorithm relies on a disjoint-set forest which represents partitions of the inheritance vectors. Initially, this forest consists of each inheritance vector in it's own set. We consider each element of the generating set of the group. For each element of the generating set, we apply that element to map each of the inheritance vectors. Let x_i be mapped to x_j by the group element. Then union of x_i and x_j is performed in the disjoint-set forest. Once we have considered each element of the generating set, and applied it to all of the inheritance vectors, the disjoint-set forest contains exactly the sets corresponding to the orbits of the group. This is easily seen by the fact that only inheritance vectors x_i and x_j appear in the same orbit if and only if there is a group element g such that $g(x_i) = x_j$. An alternative way to compute the number of orbits is to apply Burnside's

Counting Theorem [25]. However this approach requires either summing over the group elements or the state-space, and either approach is exponential.

We conjecture that an exponential algorithm is required to find the orbits of an isometry group, the efficiency in the running-time of the algorithm for finding the isometries is lost when we generate the ensemble states. Meaning, we conjecture that the problem of finding the orbits is NP-hard. Certainly computing the transition probabilities requires enumeration. This leads us to considering exponential algorithms for finding groups with the desired properties. Particularly since simulations will demonstrate that such algorithms can yield exponential improvements in the required state-space, whereas the Geiger, et al. isometry approach seems to yield only linear improvements in the size of the state-space (from simulations, data not shown).

Example

It is necessary to give an example showing that isometry groups do not yield maximal reductions in the size of the state-space. While it is true that a maximal group of isometries yields state-space savings, groups of permutations on the vertices of the hypercube give the maximal possible state-space reduction (i.e. the minimal number of orbits of any group acting on the inheritance vectors).

For example, given 4 meioses for two half-cousins, A and B, with one shared grandparent, their common grandparent and their respective parents who are half-siblings, we have 16 hypercube vertices (see Figure 4.3.2). Our individuals of interest are $I = \{A, B\}$. The IBD classes of interest are $J_1 = \{A_p\}\{A_m, B_m\}\{B_p\}$ and $J_2 = \{A_p\}\{A_m\}\{B_m\}\{B_p\}$, since these are the only partitions of alleles of individuals I that have non-empty IBD classes. The IBD classes induced on the hypercube vertices are: $D_{J_1} = \{1001, 1111\}$ and $D_{J_2} = \mathcal{H}_n \setminus D_{J_1}$.

Notice that in this instance we cannot use the IBD classes $D_J \;\; \forall J$ as the state space of a new Markov chain. For example, if we were to let Z_t be a Markov chain on the partition given by the IBD classes, then the Markov criteria would fail to hold. Specifically, consider state $X_1 = 0001$ and $x_2 = 0011$. Then by checking Equation (4.2), we have $\sum_{y \in D_{J_1}} Pr[X_t = y|X_t = 0001] = \theta(1-\theta)^3 + \theta^3(1-\theta)$ but $\sum_{y \in D_{J_1}} Pr[X_t = y|X_t = 0011] = 2 \cdot \theta^2(1-\theta)^2$.

The largest partition of $\mathcal{H}_n$ that satisfies the Markov criteria is $P_J = \{1001, 1111\}$, $P_R = \{0010, 0100\}$, $P_G = \{1011, 1101\}$, $P_B = \{0000, 0110\}$, $P_K = \{0011, 0101, 1010, 1100\}$, and $P_L = \{0001, 0111, 1000, 1110\}$. Let H be the matrix of pair-wise Hamming distances between all the vertices of the hypercube (matrices given in Fig 4.7). Then the transition probabilities take the form:

$$\frac{1}{|W_i|} Pr[Y_{t+1} = P_j | Y_t = P_i] = \sum_{y \in P_j} \theta^{|x-y|}(1-\theta)^{n-|x-y|}.$$

For example, $Pr[Y_{t+1} = P_L | Y_t = P_K] = \frac{1}{2}\theta(1-\theta)^3 + \frac{1}{2}\theta^3(1-\theta)$.

Notice that this partition cannot be the orbits of a group of isometries, because there is no $T = \phi_a \pi$ that maps $P_K \to P_K$ without also mapping some elements $P_J \to \neg P_J$. For example, $\pi = (2\ 3)$ maps $0011 \to 0101$ and $1010 \to 1100$ while ϕ_{1001} maps $0011 \to 1100$ and $0101 \to 1010$, but ϕ_{1001} maps $1001 \to 0000$, thereby violating the IBD class.

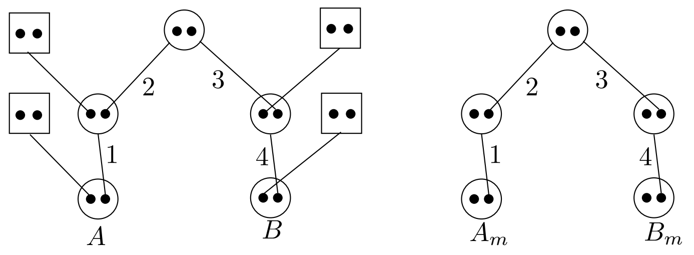


Figure 4.6: **Two Half-Cousins.** (Left Panel) A pedigree with four non-founders of which two are half-cousins together with their common grandparent. Circles and boxes represent female and male individuals, while the two black dots for each person represent their two chromosomes or alleles. Edges are implicitly directed downward from parent to child. The alleles of each individual are ordered, so that the left allele, or paternal allele, is inherited from the person's father, while the right, maternal allele is inherited from the mother. The two cousins are labeled A and B. It is easy to see that the only possible IBD is between alleles A_m and B_m, the maternal alleles of individuals A and B, respectively. (Right Panel) This makes the four male founders irrelevant to the question of IBD. The four meioses are labeled in the order of their bits, left-to-right,and the inheritance states are represented in binary as $b_1b_2b_3b_4$. Let $b_i = 0$ if that allele was inherited from the parent's paternal allele, and $b_i = 1$ if from the maternal allele. For instance, A and B are IBD only for inheritance states 1001 and 1111.

The sub-partition of the above partition that satisfies the isometry distance property is $\{1001, 1111\}$, $\{0010, 0100\}$, $\{1011, 1101\}$, $\{0000, 0110\}$, $\{0011, 0101\}$, $\{1010, 1100\}$, $\{0001, 0111\}$, and $\{1000, 1110\}$. Notice that this last partition is the orbits of the isometry group $G = \{e, \pi\}$ where e is the identity map and $\pi = (2\ 3)$ permutes the second and third bits.

Maximal Ensemble Algorithm

Consider the IBD classes, D_J for all J of interest. Of course the IBD classes are disjoint. Consider the $(2^n)!$ permutations on the vertices of the hypercube. Naively, these are all candidate permutations for our group. However, we are interested in the permutation group(s) that give us the largest orbits, alternatively the largest partition of the hypercube vertices, that satisfy the Markov property. Notice that this objective function is different than that employed by Browning and Browning, since they were looking for the largest group of isometries. However, this objective function is well motivated by the exponential algorithms that seem to be required for finding the orbits and the transition functions of isometry groups. In addition, as shown by the example, the maximal group of isometries does not yield the optimal state space reduction.

We find the optimal state-space reduction by recursively partitioning the partition, first according to the sum of the powers and then according to the powers themselves. This is possible since the Markov property must produce a partition that is a sub-partition of the IBD classes (i.e. in order to respect the IBD classes). This approach will at worst produce a

$P_R \setminus P_J$	1001	1111
0010	3	3
0100	3	3

$P_G \setminus P_R$	0010	0100
1011	2	4
1101	4	2

$P_G \setminus P_J$	1001	1111
1011	1	1
1101	1	1

$P_B \setminus P_R$	0010	0100
0000	1	1
0110	1	1

$P_B \setminus P_J$	1001	1111
0000	2	4
0110	4	2

$P_B \setminus P_G$	1011	1101
0000	3	3
0110	3	3

$P_J \setminus P_K$	0011	0101	1010	1100
1001	2	2	2	2
1111	2	2	2	2

$P_J \setminus P_L$	0001	0111	1000	1110
1001	1	3	1	3
1111	3	1	3	1

$P_R \setminus P_K$	0011	0101	1010	1100
0010	1	3	1	3
0100	3	1	3	1

$P_R \setminus P_L$	0001	0111	1000	1110
0010	2	2	2	2
0100	2	2	2	2

$P_G \setminus P_K$	0011	0101	1010	1100
1011	1	3	1	3
1101	3	1	3	1

$P_G \setminus P_L$	0001	0111	1000	1110
1011	2	2	2	2
1101	2	2	2	2

$P_B \setminus P_K$	0011	0101	1010	1100
0000	2	2	2	2
0110	2	2	2	2

$P_B \setminus P_L$	0001	0111	1000	1110
0000	1	3	1	3
0110	3	1	3	1

$P_L \setminus P_K$	0011	0101	1010	1100
0001	1	1	3	3
0111	1	1	3	3
1000	3	3	1	1
1110	3	3	1	1

Figure 4.7: Hamming distances between each pair of partition sets.

partition with each element in its own set. Since the recursive sub-partitioning at minimum splits sets in two, the number of iterations required is $O(n)$. Checking the Markov properties requires $O(4^n)$ time for each iteration, since we have to check the $2^n \times 2^n$ matrix of distances, or sums of distances, between partition elements. So, the total running time is $O(n4^n)$.

Lemma 4.3.1. *(Correctness Proof.) This partitioning algorithm produces the partition respecting the IBD classes and the Markov property which has the least number of sets.*

Proof. Clearly the partitioning algorithm produces a partition that respects the IBD, since it begins with the partition given by the IBD classes and sub-partitions it. The algorithm also produces partitions that respect the Markov property, since it iteratively sub-partitions of the IBD-class partition until the Markov property is satisfied. Since partition sets are only divided if they violate the Markov property, the algorithm necessarily finds the optimal partition. □

4.3.3 Simulation Results

We simulated pedigrees under a Wright-Fisher model with monogamy where each pair of monogamous individuals has Poisson-distributed number of offspring. There are n individuals per generation and λ is the mean number of offspring per monogamous pair. The individuals of interest, I, were the extant individuals, i.e. those in the most recent generation. These pedigrees have no inter-generational mating due to how the Wright-Fisher model is defined. To get a half-sibling pedigree, each edge of the pedigree had 50% chance of have a new parent drawn at random. Since monogamy was not preserved during this random process, the resulting pedigree had half-siblings.

Running the simulation process and the maximal ensemble algorithm 100 times for each type of simulation produced Figure 4.3.3. The maximal ensemble algorithm produces exponential reductions in the size of the state-space. Whether the relationships have half-siblings seems not to influence the practical applicability of the algorithm.

In practice, the algorithm seems limited to pedigrees of roughly 14 meioses, due to the exponential nature of enumerating the state-space. Since it is NP-hard to find inheritance-path representatives for the orbits of a group of isometries, it seems that enumeration of the state-space is unavoidable. Additionally, determining the transition rates require enumeration. Given these constraints and the practical success of the maximal ensemble algorithm, we recommend that the maximal ensemble algorithm be employed for state-space reduction, even over the Geiger, et al. $O(n^2)$ algorithm.

4.3.4 Summary

Even though past efforts at state-space reduction have focused on finding groups of isometries on the edges of the pedigree graph, it seems clear that efforts should focus on finding groups of permutations of the inheritance vectors. Even though a group of isometries operates on the edges of the pedigree graph, and would seem to be more efficient, it requires enumeration of the inheritance vectors to obtain the orbits. Furthermore, computation

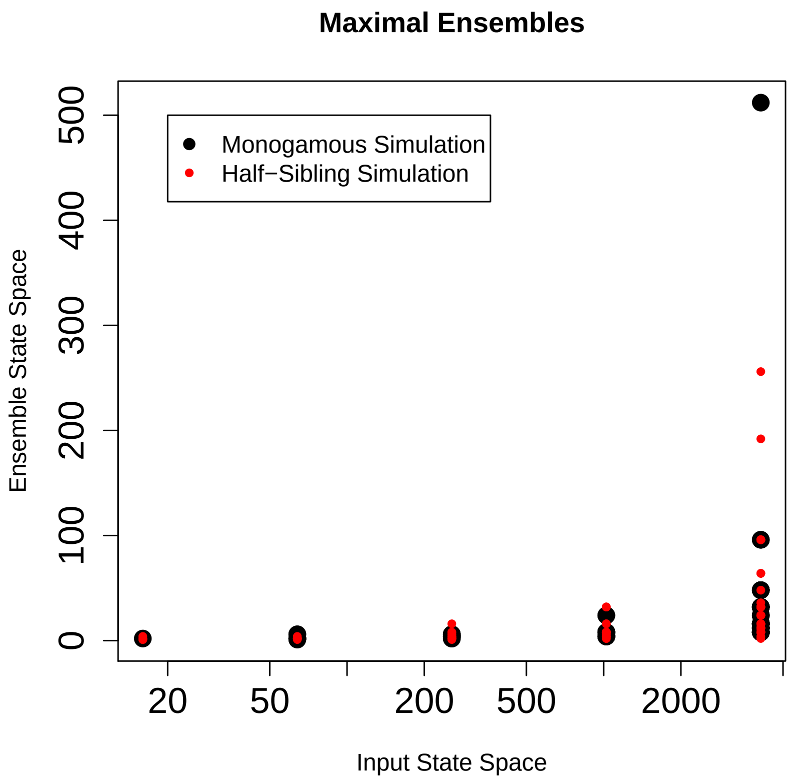


Figure 4.8: **Maximal Ensemble Algorithm Results.** The y-axis is the original size of the state space, and the x-axis give the number of ensemble states produced by the maximal ensemble algorithm. All simulated pedigrees had two generations, $n = 6$ individuals per generation, and Poisson mean $\lambda = 2$. The black points are for monogamous pedigrees and the red points are for pedigrees with half-siblings.

of the transition probabilities according to Equation 4.1 seems to require enumeration of the inheritance vectors. Therefore, it is not a disadvantage to consider using exponential algorithms to find the maximal state-space reduction.

We demonstrate that maximal state-space reduction is obtained by considering permutations of the inheritance vectors which is equivalent to considering partitions of the inheritance vectors. By doing this, in practice, we can obtain exponential reductions in the state-space required for an HMM likelihood calculation.

There are several open problems of interest. First, the computational complexity of the optimal partition problem is open. Second, the computational complexity of the maximal isometry group is also open. Third, another open problem is the computational complexity of finding the transition rates after having determined the partition of the state space. Although these problems intuitively seem NP-hard, it is unclear whether there are approximation algorithms or polynomial-time algorithms for special cases.

Chapter 5

Algorithms for Pedigree Reconstruction

Can we find the family trees, or pedigrees, that relate the haplotypes of a group of individuals? Collecting the genealogical information for how individuals are related is a very time-consuming and expensive process. Methods for automating the construction of pedigrees could stream-line this process. While constructing single-generation families is relatively easy given whole genome data, reconstructing multi-generational, possibly inbred, pedigrees is much more challenging.

This chapter introduces a theoretical pedigree reconstruction method which essentially gives an alternative definition for a pedigree in terms of descendant individuals rather than parent-child relationships. This reconstruction method is currently of conceptual value due to the inavailability of data describing descendant relationships.

In order to evaluate any pedigree reconstruction method for accuracy, we need to compare two pedigrees, a correct one and an estimated one, on the same set of extant individuals where the ancestral individuals are unobserved and therefore unlabeled. For example, perhaps only the most recent generation has genetic data. There are two natural formulations for comparing pedigrees: pedigree isomorphism and pedigree edit distance; this chapter discusses both problems.

Then taking a more practical perspective, this chapter discusses reconstructing monogamous, regular pedigrees, where pedigrees are regular when individuals mate only with other individuals at the same generation. This chapter introduces two multi-generational pedigree reconstruction methods: one for inbreeding relationships and one for outbreeding relationships. In contrast to previous methods that focused on the independent estimation of relationship distances between every pair of typed individuals, here we present methods that aim at the reconstruction of the entire pedigree. We show that both our methods outperform the state-of-the-art and that the outbreeding method is capable of reconstructing pedigrees at least six generations back in time with high accuracy.

5.1 Introduction

Genealogical methods for producing pedigree graphs, or family trees, from birth records are very expensive. To address this issue, the pedigree reconstruction problem was introduced by Thompson [95] as follows: given genetic data for a set of extant individuals, reconstruct relationships between those individuals that may involve unobserved ancestors.

Manual methods for constructing human pedigree graphs are very tedious. It requires careful examination of genealogical records, including marriage records, birth dates, death dates, and parental information found in birth certificates. Medical researchers then must carefully check records for consistency, for instance making sure that two married individuals were alive at the same time and making sure that children were conceived while the parents were alive. Genealogical methods for constructing pedigrees can involve multiple sources of information with contradictory or missing information. For example, it has been estimated that between 2-10% of people do not know who their biological father is [46, 84]. This manual process of constructing pedigrees is very time consuming. Despite the care taken, there are sometimes mistakes [8, 67, 90].

For constructing non-human pedigrees, of diploid organisms, it is often impossible to know the pedigree graph since there are no genealogical records [6, 10]. In this case it is particularly important to develop methods of automatically generating pedigrees from genomic data.

The problem of reconstructing pedigrees from haplotype or genotype data is not new. The oldest such method that the author knows of is due to Thompson [95]. That approach is essentially a structured machine learning approach where the aim is to find the pedigree graph that maximizes the probability of observing the data under the pedigree model, also called the likelihood of the pedigree. (That approach is directly analogous to maximum likelihood methods for phylogenetic reconstruction which also try to find the phylogenetic tree that maximize the likelihood.) Notice that this method reconstructs both the pedigree graph and the ancestral haplotypes which is a very time-consuming step. Thus, this approach is limited to extremely small families, perhaps 8 people total, since the algorithms for computing the likelihood of a fixed pedigree graph are exponential [61] and there are an exponential number of pedigree graphs to consider [91].

The current state-of-the-art method is an HMM-based approximation of the number of meioses separating a pair of individuals [87]. This approach dispenses with any attempt to infer haplotypes of ancestral individuals, and instead focuses on the number of generations that separate a pair of individuals. In this approach the hidden states of the HMM represent the identity-by-descent (IBD) of a pair of individuals. Two individuals are identical-by-descent for a particular allele if they each have a copy of the same ancestral allele. The probability of the haplotype data is tested against a particular type of relationship. The main draw-back of this approach is that it may estimate a set of pair-wise relationships that are inconsistent with a single pedigree relating all the individuals.

Thatte and Steel [92] examined the problem of reconstructing arbitrary pedigree graphs from a synthetic model of the data. Their method used an HMM model for the ancestry of each individual to show that the pedigree can be reconstructed only if the sequences are sufficiently long and infinitely dense. Notice that this chapter uses an unrealistic model of recombination where every individual passes on a trace of their haplotypes to all of their descendants. Kirkpatrick [49] introduced a more simple, more general version of the reconstruction algorithm introduced by Thatte and Steel.

Attempts to construct sibling relationships are known to be NP-hard, and attempts to infer pedigrees by reconstructing ancestral haplotypes are be NP-hard. Two combinatorial versions of the sibling relationship problem were proven to be NP-hard, both whole- and half-sibling problem formulations [6, 83]. If ancestral haplotypes are reconstructed in the process of inferring a pedigree, as in Thompson's structured machine learning approach, then the inheritance probabilities of data must be computed on the pedigree graph. For instance, we might want to compute the likelihood, or the probability of observing the data given inheritance in the pedigree. This calculation is NP-hard for both genotype [76, 62] and haplotype [48] data. This means that any efficient pedigree reconstruction method will need to find ways to avoid both these hardness problems.

The novel contributions of this chapter are two-fold. First, we introduce a method for evaluating the accuracy of pedigree reconstruction methods. This is important, because existing evaluation methods cannot detect small differences between pedigrees. Second, we introduce two practical pedigree reconstruction algorithms for genotype data.

Evaluating any reconstruction method requires running the method on an instance in which the true pedigree is known and then comparing the results of the method to the known true pedigree. Both the estimated pedigree and the true pedigree will have the same set of extant individuals, but each may contain a different set of inferred ancestors. Since there are no data to uniquely label the inferred ancestors, there is no way to directly compare the estimated pedigree to the true one. As a result, the topology of the two pedigrees must be compared in a fashion that respects the labels of the extant individuals.

There are two natural formulations for the pedigree comparison problem: pedigree isomorphism and pedigree edit distance. While the isomorphism problem only identifies pairs of pedigrees whose topologies match exactly, the pedigree edit distance allows differences between the pedigrees. Both isomorphism and edit distance are discussed in this chapter.

The contribution of this chapter to pedigree reconstruction is two algorithms that avoid the exponential likelihood calculations. We do this by specifically *not* reconstructing ancestral haplotypes and by *not* trying to optimize sibling groups. We use estimates of the length of genomic regions that are shared identical-by-descent. In two related individuals, a region of the genome is identical-by-descent (IBD) if and only if a single ancestral haplotype sequence was the source of the sequence inherited in the two individuals. The length of IBD regions gives a statistic that accurately detects sibling relationships at multiple generations. We have two algorithms: one for constructing inbred pedigrees (CIP) and one for constructing outbred pedigrees (COP). For our outbreeding algorithm the statistic is testable in polynomial time. For our inbreeding algorithm, the statistic is computable in time dependent on the number of meioses in the predicted pedigree. Our outbreeding method works to reconstruct at least six generations back in time. Both methods are more accurate than the state-of-the-art method by Stankovich et al. [87].

5.2 Pedigree Structure and a Simple Reconstruction Algorithm

The work in this section was a novel contribution by the author [49] also appearing in [54].

An alternative formulation of a pedigree would allow the hypothesis that a set of individuals is descended from a common ancestor (called a **descendant split**), without specifying the number of generations between each of the individuals and their common ancestor(s). The presence or absence of a single descendant hypothesis may only change the closeness of the relationship between a pair of individuals (perhaps from cousins to 2nd-cousins), rather than removing the relationship entirely. This is in contrast to the traditional formulation of a pedigree as a collection of parent-offspring edges, where a missing edge entirely changes the nature of many relationships.

Definition 5.2.1. *Let I be the set of individuals in a pedigree, and let X be a set of labeled individuals. The* **descendant split** *(or* **d-split***) of an individual $i \in I$ is defined as a subset of X:*

$$D_i(X) \quad = \quad \{j \in X | \;\; j \textit{ is descended from } i\}$$

where every individual is considered a descendant of itself. For a particular set of interest, X, refer to the set of d-splits as $\mathcal{D}_X = \{D_i(X) \mid i \in I\}$.

Each d-split specifies some relationship between all the individuals in D_i. For the example given in the left panel of Figure 5.1, when all individuals are labeled the list of d-splits, $\mathcal{D}_I$, are: $D_1(I) = \{1\}$, $D_2(I) = \{2\}$, $D_3(I) = \{1,2,3\}$, $D_4(I) = \{1,2,4\}$, $D_5(I) = \{1,2,3,5\}$, $D_6(I) = \{1,2,3,6\}$, $D_7(I) = \{1,2,4,7\}$, $D_8(I) = \{1,2,4,8\}$, $D_9(I) = \{1,2,3,4,5,6,7,8,9\}$, and $D_{10}(I) = \{1,2,3,4,5,6,7,8,10\}$. Similarly, if we restricted our attention to $X = \{1,2\}$, then $\mathcal{D}_X$ would contain $\{1\}$, $\{2\}$, and $\{1,2\}$.

The term "descendant split" is deliberately chosen to evoke the image of a split in a perfect phylogeny and to pay homage to phylogenetic reconstruction methods [82] which are a source of inspiration for this work. Just as a set of splits determines a class of perfect phylogeny trees that are compatible with the splits, a set of descendant splits specifies a class of pedigree graphs that are compatible with the splits. We will formalize this idea with several lemmas.

Lemma 5.2.1. *Let $\mathcal{D}_I = \{D_i(I) \mid i \in I\}$ be the d-splits defined by a pedigree P. This set can be used to construct a unique pedigree which is identical to pedigree P.*

Lemma 5.2.2. *Let $\mathcal{D}_X = \{D_i(X) \mid i \in I\}$ be the d-splits defined by a pedigree P and a set X. This set of d-splits specifies a class of pedigrees compatible with the splits. Pedigree P is one of the pedigrees compatible with the d-splits.*

Consider the d-splits in $\mathcal{D}_I$. Any singleton d-split, $D_i(I) \in \mathcal{D}_I$ with $|D_i(I)| = 1$, represents an individual that is childless. Therefore these d-splits represent individuals in the most recent generation of the pedigree. Now, find some ancestor i_1 and examine any directed path descending from that individual, for example, $i_1 \rightarrow i_2 \rightarrow ... \rightarrow i_{k-1} \rightarrow i_k$, where the arrow indicates a directed parent-offspring relationship. We see that the d-splits along that descendant path are ordered $D_{i_1} \supset D_{i_2}(I) \supset ... \supset D_{i_k}(I)$. Indeed, the cardinality of the d-split sets $D_{i_j}(I)$ strictly decreases as we consider individuals lower in the path. These two ideas result in a simple algorithm for constructing the pedigree.

Algorithm 3 GraphConstruction()

```
1: $Heap := (D_{i_0}, ..., D_{i_k})$ where $|D_{i_0}| \leq |D_{i_1}| \leq ... \leq |D_{i_k}|$
2: Create pedigree $P$ with nodes $\{i_0, i_1, ..., i_k\}$.
3: while $Heap \neq \emptyset$ do
4:    $D_{i_j} := pop(Heap)$
5:    Look for the smallest $D_{i_f}$ and $D_{i_m}$, respectively the female and male splits, such that
      $D_{i_j} \subseteq D_{i_f}$ and $D_{i_j} \subseteq D_{i_m}$
6:    if $D_{i_f}$ and $D_{i_m}$ are found then
7:       add to $P$ the edges $i_m \rightarrow i_j$ and $i_f \rightarrow i_j$
8:    else
9:       $i_j$ is a founder and has no parents.
10:   end if
11: end while
12: return $P$
```

Lemma 5.2.3. *Let $\mathcal{D}_I = \{D_i(I) \mid i \in I\}$ be the d-splits defined by a pedigree P. This set can be used to construct a unique pedigree which is identical to pedigree P.*

Proof. of Lemma 5.2.1
Since we have a d-split for every individual, the algorithm will either assign founder status or parents to every individual. Now, if we look at a single step in the algorithm, each individual will be assigned the parents. Due to the strictly increasing cardinality of d-splits as one moves up the pedigree (notice that this is due to an individual being contained in their own d-split), these parents are represented by the unique two smallest descendant splits. □ □

Example. If we take the d-splits $\mathcal{D}_I$ from pedigree in the left panel of Figure 5.1, we can apply the algorithm to construct the pedigree. Figure 5.1 shows the d-splits using a Venn diagram. Each step in the algorithm constructs a set of parent-child edges. This example also illustrates the ambiguity of the d-splits when individuals in the pedigree are unlabeled. As noted before, if only individuals $\{1, 2\}$ are labeled, then many of the ancestral d-splits are identical.

We can use the same algorithm when we consider d-splits on a subset of the individuals, $X \subset I$. As long as we have a separate d-split for each individual in the pedigree, we will know the number of generations in each lineage because all the parents at all generations will be represented. The main difference is that each descendant path has *non-decreasing*

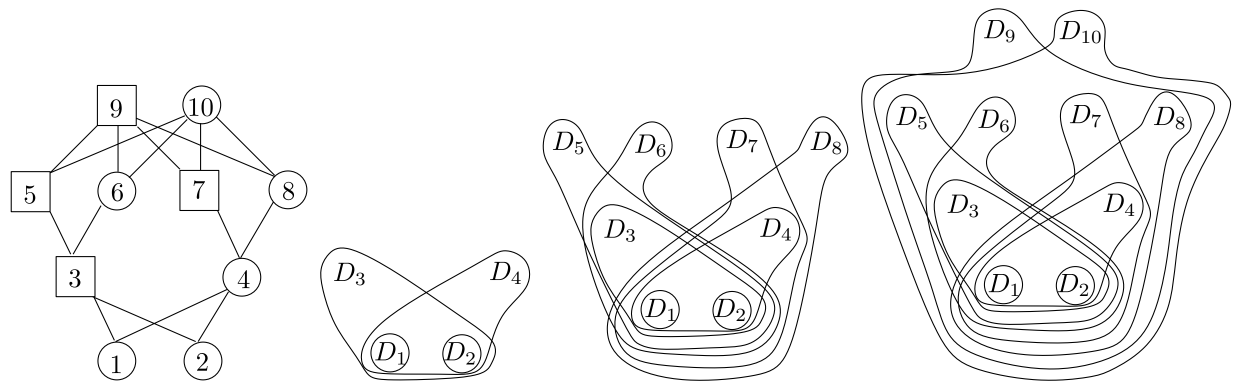


Figure 5.1: **Constructing a Pedigree from the Full D-Splits.** Given the d-splits in $\mathcal{D}_I$ for the set I of all the individuals in the pedigree on the left, we can use the construction algorithm to recover the pedigree. These are the Venn diagrams of the construction at three different steps in the algorithm. Each panel shows a full generation of algorithm iterations, and right-most panel shows the complete construction. Each d-split is drawn as a set containing the related individuals. Each set in the diagram is labeled with the name of its d-split, and the names of the d-splits are arbitrary as long as they are distinct.

cardinality of d-splits as we move backwards in time. The missing information, now, is in not knowing which d-split was generated by the parent versus a more distant ancestor. For the example we gave above, if $X = \{1, 2\}$, then $D_9(X)$ and $D_7(X)$ are indistinguishable.

Lemma 5.2.4. *Let $\mathcal{D}_X = \{D_i(X) \mid i \in I\}$ be the d-splits defined by a pedigree P and a set X. This set of d-splits specifies a class of pedigrees compatible with the splits. Pedigree P is one of the pedigrees compatible with the d-splits.*

Proof. of Lemma 5.2.2
Again, since we have a d-split for every individual, the algorithm will either assign founder status or parents to every individual. Now, if we look at a single step in the algorithm, each individual will be assigned some parents, due to the non-decreasing cardinality of d-splits as we consider d-splits for individuals in older generations. However, the construction will be different for re-orderings of the d-splits. This means that we may not be able to resolve the correct labels for individuals $I \setminus X$ in the interior of the pedigree. Even though we cannot resolve the interior, if we run the algorithm once for each possible ordering of the d-splits, we obtain a collection of pedigrees that are consistent with the d-splits. The actual pedigree will be contained in this collection. □ □

Notice that this idea of descendant splits is a general property of any directed acyclic graph with fixed in-degree. This idea provides a core description of the pedigree reconstruction algorithm that Thatte and Steel proposed [92]. They proposed an artificial generative model of inheritance in which every individual contributed some unique alleles with high probability. These unique alleles then became markers for each d-split, allowing them to piece together the lineages of multiple leaf individuals.

5.3 Accuracy Measurements

The work in this section appears in [54].

Evaluating any reconstruction method requires running the method on an instance in which the true pedigree is known and then comparing the results of the method to the known true pedigree. Both the estimated pedigree and the true pedigree will have the same set of extant individuals, but each may contain a different set of inferred ancestors. Since there are no data to uniquely label the inferred ancestors, there is no way to directly compare the estimated pedigree to the true one. As a result, the topology of the two pedigrees must be compared in a fashion that respects the labels of the extant individuals.

There are two natural formulations for the pedigree comparison problem: pedigree isomorphism and pedigree edit distance. While the isomorphism problem only identifies pairs of pedigrees whose topologies match exactly, the pedigree edit distance allows differences between the pedigrees. Both isomorphism and edit distance are discussed in this section.

5.3.1 Isometry between Pedigrees

For a certain class of pedigrees and individuals of interest, we can detect the isomorphism of two pedigrees in $O(n^2 \cdot |X|^2)$ time where $n = |I|$. Following from the previous section, there are certain instances where a pedigree and its individuals of interest, $X \subseteq I$ with $|X| \leq |I|$, yield d-splits that uniquely determine the pedigree, rather than determining a class of pedigrees, as in Lemma 5.2.2.

Definition 5.3.1. *A pedigree P and its individuals of interest X is* **resolvable** *from its d-splits if and only if every individual has a d-split that is either distinct from the d-splits of all other individuals or identical to at most one other individual, who is the mating partner in a monogamous pairing.*

Lemma 5.3.1. *Two resolvable pedigrees are isomorphic if and only if both pedigrees contain the same list of descendant-splits.*

Proof. Here we simply use Algorithm 3 and lean on the proof of Lemma 5.2.1. Notice that in Lemma 5.2.2 different orderings of the d-splits yield different pedigrees from the Graph-Construction() algorithm. By definition, a resolvable pedigree is one for which there is a single non-decreasing ordering of the d-splits up to re-orderings of the splits for monogamous mates. This means there is an ordering of the splits such that we can reconstruct a unique pedigree graph from the list of splits, by Algorithm 3. Therefore, two graphs are isomorphic if and only if they have the same d-splits and there is a unique pedigree which can be constructed from the d-splits. □

The algorithm for detecting isomorphic pedigrees proceeds to check the condition given in the definition followed by the condition given in the lemma. Checking resolvability requires checking equality of two descendant-splits, an order $|X|^2$ operation. Since there are at most n d-splits which need to be compared to each other, these operations have running time $O(n^2 \cdot |X|^2)$ where n is the number of individuals in the pedigree.

5.3.2 Edit Distance between Pedigrees

In the previous section, we presented a method for determining whether certain pedigrees are isomorphic. However, we are interested not only in determining whether two pedigrees are isomorphic, but in how close they are to being isomorphic. Such a distance measure would be useful for evaluating pedigree reconstruction methods; for example, it would allow us to determine how good a predicted pedigree is by comparing it to a correct pedigree graph.

Informally, given two arbitrary pedigree graphs, $P = (I(P), E(P))$ and $Q = (I(Q), E(Q))$, we want to find the minimum number of parent-child edge additions/deletions required to convert Q into P. We call this the *edge edit distance* between P and Q. Notice that it is not necessary that $|I(P)| = |I(Q)|$ because addition/deletion of edge-less nodes is free.

Let us define edge edit distance more formally. For an individual $i \in I(\cdot)$, let $s(i) \in \{m, f\}$ indicate male and female gender, respectively. Define a *matching* of pedigrees *between* P and Q to be a subset M of $I(P) \times I(Q)$ such that $(i, j) \in M$ only if $s(i) = s(j)$, and for each i (respectively, j), there is at most one j (respectively, i) such that $(i, j) \in M$. Given such a matching M and some $i \in I(P)$, we define $M(i) = j$ if $(i, j) \in M$ and $M(i) = \lambda$ otherwise, where λ is a special symbol reserved for nodes that are not matched. We will abuse notation and use $M(j)$ to denote the analogous function for individuals in $I(Q)$ as well. With this notation, we define the *match distance* incurred by M as

$$d(M) = d_{P,Q}(M) + d_{Q,P}(M),$$

where $d_{G,G'}(M) = |\{(i, j) \in E(G) | (M(i), M(j)) \notin E(G')\}|$. Here, $d_{G,G'}(M)$ is the number of parent-child edges in the pedigree G that do not correspond under the matching M to parent-child edges in G'.

Now, we can define a distance between pedigrees. Let the *edit distance* between pedigrees P and Q be defined as the minimum matching distance:

$$D_{P,Q} = \min_M d(M).$$

When we have a set of labeled individuals X which is a subset of both $I(P)$ and $I(Q)$ (recall that this is a set of distinguishable individuals for which we may have data, for example), we can force the distance to respect the labeled set by minimizing only over those matchings for which $(i, i) \in M$ for all $i \in X$. Notice that the dual optimization problem is that of maximizing the number of matched edges. When convenient, we occasionally deal with the dual.

The problem of finding the pedigree distance is a specific case of the well-studied problem of inexact graph matching [19]. Inexact graph matching is not only NP-hard in general, but MAX SNP-hard even when restricted to trees [103], and there is a extensive literature on heuristics and algorithms for inexact graph matching. However, the hardness results do not apply to pedigree distance—perhaps the hard cases of inexact graph matching are non-pedigrees—and the algorithms do not take advantage of the structure of pedigrees. In Section 4, we show that computing pedigree distance is APX-hard, and in Section 5, we present several heuristics and algorithms for calculating pedigree distance in general and in specific cases.

Hardness of Computing Pedigree Edit Distance

In this section, we show that calculating the edge edit distance between two pedigrees is APX-hard for monogamous out-bred pedigrees (i.e. tree-like pedigrees), by reducing from Minimum Common Integer Partition. For clarity, when applied to trees, we call the edge edit distance *cut-and-paste* distance (cut/paste for short). A cut/paste operation involves deleting (cutting) an edge $x \to y$ and adding (pasting) a new edge $z \to y$.

Given two minimization problems Y and Z, an *L-reduction* from Y to Z is a pair of polynomial-time functions f, g and a pair of positive constants α, β meeting the following conditions.

(1) For every instance y of Y, $f(y)$ is an instance of Z with

$$opt(f(y)) \leq \alpha \cdot opt(y),$$

(2) For a feasible solution z to $f(y)$, $g(z)$ is a feasible solution to y such that

$$|opt(y) - val(g(z))| \leq \beta \cdot |opt(f(y)) - val(z)|.$$

Note that $opt(y)$ is the value of the optimal solution to an instance y, while $val(z)$ denotes the value of solution z. With the above two properties, it is easily seen that the following inequality on the relative errors of approximation holds:

$$\frac{|opt(y) - val(g(z))|}{|opt(y)|} \leq \alpha\beta \cdot \frac{|opt(f(y)) - val(z)|}{|opt(f(y))|}.$$

When the above conditions are satisfied, if it is NP-hard to approximate Y with a factor of $1 + \alpha\beta\epsilon$, then it is NP-hard to approximate Z with a factor of $1 + \epsilon$. In particular, if

We reduce MCIP (Minimum Common Integer Partition) to MCPDT (Minimum Cut/Paste Distance between Trees) with an L-reduction. Notice that the MCPDT is the pedigree edit distance viewed as cut-and-paste operations on trees.

A partition of an integer n is a multiset $\{n_1, n_2, ..., n_t\}$ such that $\sum_{1 \leq i \leq t} n_i = n$. For example, when $n = 8$, $\{3, 2, 2, 1\}$ is a partition of n.

A partition of a multiset $S = \{x_1, x_2, ..., x_p\}$ is a multiset union of all the partitions $P(x_i)$, i.e., $\cup_i P(x_i)$. A multiset X is a common partition of two multisets $S_1 = \{x_1, x_2, ..., x_p\}, S_2 = \{y_1, y_2, ..., y_q\}$ if there are partitions P, Q with $\cup_i P(x_i) = \cup_j Q(y_j) = X$. For example, given $S_1 = \{8, 5\}, S_2 = \{9, 4\}$, $X = \{5, 3, 2, 2, 1\}$ is a common partition of S_1, S_2.

MCIP (Minimum Common Integer Partition)

Instance: Two multisets of integers S_1, S_2, integer k.

Question: Do S_1, S_2 admit a common partition of size k?

It was shown in [17] that MCIP is APX-hard.

MCPDT (Minimum Cut/Paste Distance between Trees)

Instance: Two directed rooted unlabeled trees T_1, T_2, integer k.

Question: Can T_1 be converted into T_2 using k cut/paste operations?

Note that the question in MCPDT is also equivalent to: Can T_2 be converted into T_1 using k cut/paste operations? Or, Can T_1, T_2 be converted into a tree T using a total of k cut/paste operations? Or, Can T_1, T_2 be cut into a common forest F each using k edge cuts?

The reduction is simple. Given $S_1 = \{x_1, x_2, ..., x_p\}, S_2 = \{y_1, y_2, ..., y_q\}$, we construct two trees T_1, T_2 with roots r_1, r_2 such that the descendants of r_1 (resp. r_2) is composed of p (resp. q) paths, each is of size x_i (resp. y_j), for $1 \leq i \leq p$ (resp. $1 \leq j \leq q$). We need to cut T_1, T_2 into a common forest, in which each tree, except the ones containing r_1, r_2, is a path.

Let $opt(MCIP)$ and $A(MCIP)$ be the values of the optimal and approximate solution of MCIP, respectively. Let min $= \min\{p, q\}$. Then the value of the optimal solution for MCPDT is $opt = opt(MCIP) - \min$. (In other words, we can conclude that S_1, S_2 has a solution of size k if and only if T_1, T_2 each can be cut into a common forest with $(k - \min)$ cuts.) Moreover, $A(MCPDT) = A(MCIP) - \min$ is the value of a feasible solution of MCPDT. Clearly we have

(1) $opt \leq \alpha \cdot opt(MCIP)$, by setting $\alpha = 1$.

(2) $|opt(MCIP) - A(MCIP)| \leq \beta \cdot |opt - A(MCPDT)|$, by setting $\beta = 1$.

To see (2), $|opt - A(MCPDT)| = |opt - (A(MCIP) - \min)| = |opt(MCIP) - A(MCIP)| \geq |opt(MCIP) - A(MCIP)|$.

Therefore, this reduction is an L-reduction. As MCIP is APX-hard, MCPDT is also APX-hard. This implies that unless P=NP, there is no way to obtain a PTAS for MCPDT. Since MCPDT is a special case of computing edge edit distance, this implies that edge edit distance is APX-hard. This proof works for labeled pedigrees provided that the labels are the same in the two pedigrees.

Algorithms for Computing Pedigree Edit Distance

We showed in the previous section that no polynomial time algorithm exists to determine the pedigree distance unless $P = NP$. Indeed, even for monogamous pedigrees, there is no PTAS unless $P = NP$. In this section, we present a polynomial time randomized heuristic for calculating pedigree distance on general labeled pedigrees, a dynamic programming algorithm which provides an optimal solution for a restricted set of pedigrees in time which depends on distance between pedigrees, and we summarize a few other algorithms for general and special cases.

Let a *regular* pedigree be one where every individual is monogamous and only individuals of the same generation mate with each other. Consider the case of two regular pedigrees both having the same set of labeled individuals, X.

Heuristic Random Matching Algorithm using D-Splits. The randomized matching algorithm for regular pedigrees is exceptionally simple. At each generation, among individuals of the same gender, choose a match-pair with probability proportional to the number of same-labeled individuals, $i \in X$, in the descendant sets of the two pedigrees. If there were separate paths from each leaf to the individual under consideration, then this algorithm would give equal weight to each path. However, these paths share edges, so it is difficult to analyze this heuristic.

This algorithm is polynomial, because at each generation it creates a $n \times n$ matrix of individual match probabilities for each gender (where there are $2n$ individuals per generation). The matches are then drawn iteratively from these probability matrices (without replacement of previously matched individuals).

Dynamic Programming Algorithm for Matched Generations. Suppose we have two regular pedigrees P and Q, each of which is made up of g generations of m males and m females each. If the pedigrees P and Q are similar enough and the youngest generations of P and Q are entirely labeled with unique labels, we can use dynamic programming to find an optimal matching between them in time exponential in the edit distance.

We will need some notation to describe the algorithm. First, let the generations of pedigree P be numbered with the youngest generation being generation 1 and the oldest generation being generation g. For a pedigree P and a subset $S \subseteq 1, 2, ..., g$, let $P|_S$ denote the sub-pedigree containing the i-th generation of P for every $i \in S$. Let $M(S)$ be the set of all matchings from $P|_S$ to $Q|_S$. For $M \in M(S)$ and i such that $i \geq j$ for all $j \in S$, let $B_i(M)$ be the edit distance between $P|_{\{1,\ldots,i\}}$ and $Q|_{\{1,\ldots,i\}}$ with the restriction that the only matchings considered when calculating the distance are those that agree with M where M is defined. We will abuse notation a bit by letting $M(i)$ denote $M(\{i\})$ and letting $P|_i$ denote $P|_{\{i\}}$. Finally, for matchings $M \in M(S)$ and $T \subseteq S$, let $d_T(M)$ be the edge edit distance corresponding to M restricted to $P|_T$.

Our algorithm rests on the following simple relation.

Lemma 5.3.2. *For every $i \in \{2, \ldots, g\}$, and for every $M \in M(\{i, i-1\})$, let M_{i-1} be the restriction of M to $P|_{\{i-1\}}$. Then we have*

$$B_i(M) = B_{i-1}(M_{i-1}) + d_{\{i,i-1\}}(M) \tag{5.1}$$

Lemma 5.3.2 gives rise to a simple dynamic programming algorithm. However, the problem with this straightforward algorithm is that its run-time is factorial in m, the number of males/females in each generation. For each value of i and each fixed matching M_{i-1}, there are $(m!)^2$ possible matchings M to process. Therefore, the run-time of this algorithm is $O((m!)^{2g})$.

We can improve this algorithm if we know that the two pedigrees under consideration are sufficiently similar at each generation by realizing that there is no need to consider all matchings for each generation. Suppose we are promised that the optimal matching M of P to Q has $d_{\{i,i-1\}}(M) < k$ for every $1 < i \leq g$. Then in the algorithm above we would only need to process, for each $1 < i \leq g$ and each $M_{i-1} \in M(i-1)$, the matchings $M \in M(\{i, i-1\})$ such that M restricted to $P|_{i-1}$ equals M_{i-1} and $d_{\{i,i-1\}}(M) < k$.

The enumeration of the set $\{M \in M(\{i, i-1\})\colon M|_{i-1} = M_{i-1}, d_{\{i,i-1\}}(M) < k\}$ can be done recursively by choosing a node of $P|_i$, matching it to a node in the i-th generation of Q, and then recursively doing the same to another unlabeled node in $P|_i$, all the time keeping track of the accrued cost of the matching so far and not making any assignment that will push that cost above k. The pseudo-code for this enumeration is given below.

In the pseudo-code, the function Process carries out the dynamic programming implied by the recursion in Lemma 5.3.2. It takes as input a matching M of generations i and $i-1$

Algorithm 4 EnumerateMatchings($M_i, M_{i-1}, cost$)

```
input: Access to $P, Q$, pedigrees with $g$ generations of $m$ males/females each.
input: $i$ specifies which pair of generations is getting matched.
 1: if $|M_i| = 2m$ then
 2:    Process($M_i, M_{i-1}$)
 3:    RETURN
 4: end if
 5: $u \leftarrow$ a node of $P|_i$ unmatched by $M_i$
 6: for $a \in Q|_i$ such that $s(a) = s(u)$ and $a$ is unmatched by $M_i$ do
 7:    add $u \mapsto a$ to $M_i$
 8:    if $d(M_i|_{\{i,i-1\}}) \leq k$ then
 9:       EnumerateMatchings($M_i, M_{i-1}, d_{\{i,i-1\}}(M_i)$)
10:    end if
11:    remove $u \mapsto a$ from $M_j$
12: end for
13: RETURN
```

that is made up of a matching M_i of the i-th generation and a matching M_{i-1} of the $i-1$-th generation. With this matching, it does the following:

1. Calculates the distance $d_{\{i,i-1\}}(M)$ on the sub-pedigrees consisting of only generations i and $i-1$
2. Retrieves $B_{i-1}(M_{i-1})$
3. Stores that $B_i(M_i)$ equals $B_{i-1}(M_{i-1}) + d_{\{i,i-1\}}(M)$

Step (1) is easily done in $O(m^2)$ time. For steps (2) and (3) to be efficient, the values of $B(-)$ must be stored efficiently. To do this, we define an ordering on the set $M(\{i, i-1\})$ by interpreting each matching as an m-digit m-ary number. We can then store the matchings and their associated costs in a list sorted according to the ordering on the matchings, and use binary search to insert and retrieve matchings while keeping the list sorted. This scheme allows the Process function to run in time $O(m)$ each time it is called.

But how many times is Process called? The following two lemmas together bound the number of matchings processed by EnumerateMatchings.

Lemma 5.3.3. *The number of matchings on which EnumerateMatchings will call the Process function is at most $T(m,k)^2$ where T is defined by the recurrence relation*

$$T(n,c) = T(n-1,c) + (n-1)T(n-1,c-2)$$

with initial conditions $T(1,c) = 1$ and $T(n,0) = T(n,1) = 1$.

Proof. First, suppose that there are only m individuals of one gender to match. Initially, EnumerateMatchings has m individuals whom it has to match and k “cost points” that it can use up. In the best case, there is only one individual in $Q|_i$ to whom u can be matched without using any cost points (increasing the cost of the matching). This follows from the case with one child, where parent and child in one pedigree are matched to parent and child in the other pedigree. Besides this option there are at most $m-1$ other options, each of which will increase the cost of the matching by at least 2 (since at least one edge will have to be deleted and one edge will have to be added). This establishes the recurrence. The initial conditions follow from the following facts: 1) when one individual is left to be matched, there is only one possible way to complete the matching being built, and 2) when the matching being built is already costing k (i.e. there are 0 cost points left), there is at best only one way to complete the matching.

This bounds the number of matchings of each gender by $T(m,k)$. Since this process occurs independently for each gender, the number of total matchings is at most $T(m,k)^2$. □

Lemma 5.3.4. *The recurrence T defined above satisfies $T(n,c) = O(n^C)$ where C is the greatest even integer less than or equal to c.*

Proof. We proceed by induction on c. The initial conditions of T give us our base case of $c = 0$. The general case follows from bounding the difference between successive values of $T(\cdot, c)$: the recurrence gives us that $T(n,c) - T(n-1,c) = (n-1)T(n-1,c-2)$, which is $nO(n^{C-2}) = O(n^{C-1})$ by the inductive hypothesis. □

The run-time of this more efficient algorithm is dominated by its last step, in which all the matchings between the oldest generations of P and Q are considered and the best one is chosen. By lemma 5.3.4, the number of these matchings is at most $O(m^{2k(g-1)}) = O(m^{2d})$ where d is the maximum possible distance between the two pedigrees. Thus, if k (the distance between pairs of successive generations) and g (the number of generations) are small, the algorithm can efficiently find a good edit distance. If no matching has distance less than k between every two generations, the algorithm fails by returning the matching found by the random heuristic. The algorithm is described in more detail in the appendix.

Heuristic Improvements to the Dynamic Programing Algorithm. We can turn the DP algorithm into a heuristic by enumerating only some matchings that are obtained as solutions to a matching problem. For each generation and for a fixed labeling of the previous generation, similar to the two-generation matching problem, we can devise an instance of the minimum-weight perfect matching problem for which we can enumerate the k-best solutions. This can greatly improve the running-time of the EnumerateMatchings() method. For each generation i and for some matching $i-1$ of the previous generations, generate the two-generation maximum-weight bipartite matching instance for each gender with nodes $I_F(P) \cup I_F(Q)$ for the females and nodes $I_M(P) \cup I_M(Q)$ for the males. For sex S, we convert this into an instance of the minimum-weight perfect matching problem as follows. Create dummy vertices for half of the bipartite graph, where there is zero cost to match any vertex to a dummy in the opposite half of the graph. For vertices representing individuals in pedigree P we create $|I_S(Q)|$ dummy vertices which we call $D_S(Q)$, and respectively for Q we have $D_S(P)$ where $|D_S(P)| = |I_S(P)|$. So we have vertices

$V = I_S(P) \cup D_S(Q) \cup I_S(Q) \cup D_S(P)$. with weights $w_S(i,j) = 0$ for $i \in D_S(Q), j \in V$ and for $i \in V, j \in D_S(P)$. Let $m = \max_{i,j} w_S(i,j)$ be the maximum weight edge in the graph. Now the weights $x_S(i,j)$ for the minimum-weight bipartite matching instance are then $x_S(i,j) = m - w_S(i,j)$ which yields non-negative weights.

Now, we have an instance of the minimum-weight perfect matching problem which we can solve for the k-best matchings. Chegireddy and Hamacher give an algorithm for finding the k-best perfect matchings [15].

Two-Generation Polynomial-Time Algorithm. When P and Q are both two-generation pedigrees where the labeled individuals X entirely determine the matching of the most recent generation, we have a polynomial-time algorithm. We construct two maximum-weight bipartite matching instances, one for each gender, whose solutions gives us the best matching. Notice that in doing this, we deal with the dual problem to the minimum edit distance, which is the maximum number of matched edges.

Let $I_M(S)$ and $I_F(S)$ be the male and female individuals of pedigree S, respectively. Let $G^M = (I_M(P) \cup I_M(Q), E(G_M))$ be the bipartite graph for males and $G_F = (I_F(P) \cup I_F(Q), E(G_F))$ the similar graph for females.

The edge weights for the male graph are $w_M(i,j)$ with $(i,j) \in E(G_M)$ and similarly, $w_F(i,j)$ for $(i,j) \in E(G_F)$. Let $s(i)$ be the gender of individual i. For every $(i,j) \in I(P) \times I(Q)$, we will have two nodes $i \in I_{s(i)}(P)$ and $j \in I_{s(i)}(Q)$. Edges are created as follows:

1. For all $i \in X$, create edge $(i,i) \in E(G_{s(i)})$.
2. For all $(i,j) \in I(P) \times I(Q)$ where $i \notin X$, $j \notin X$ and $s(i) = s(j)$, if i and j each have at least one common child $c \in X$, create $(i,j) \in E(G_{s(i)})$ with weight $w_{s(i)}(i,j)$ equal to the number of children having the same label.

When we compute the maximum-weight matching on these graphs G_M and G_F, we will obtain the matching of the parents of the labeled individuals that minimizes the number of mis-matched child edges. Notice that labeled individuals are forced to match with the corresponding individual with the same label in the other pedigree. This holds even for P and Q which are each collections of families.

Branch and Bound Algorithm. An exact solution can be obtained by branching on all the possible matchings for each generation before continuing recursively to consider older generations. Indeed, this algorithm does not even require the generations in order to branch, since it can simply branch on every possible node-paring. We implemented a branch-and-bound algorithm that does this branching procedure and bounds the search if it becomes clear that the current matching will have a worse score than the best found so far. In the case of monogamy, we simply match monogamous couples rather than individual parents. In the case of regular pedigrees we can bound the search by finding the best possible matching via the two-generation algorithm. For each generation i and for some matching of the previous generations, construct the two-generation maximum-weight bipartite matching instance and solve it. The solution yields the maximum number of edges that can be shared by any pairing of the nodes at generation i given the existing node-pairings for generation $i-1$. This number can be used to bound the search and by not following branches that are certainly have worse matches than the best matching found so far.

Simulation Results

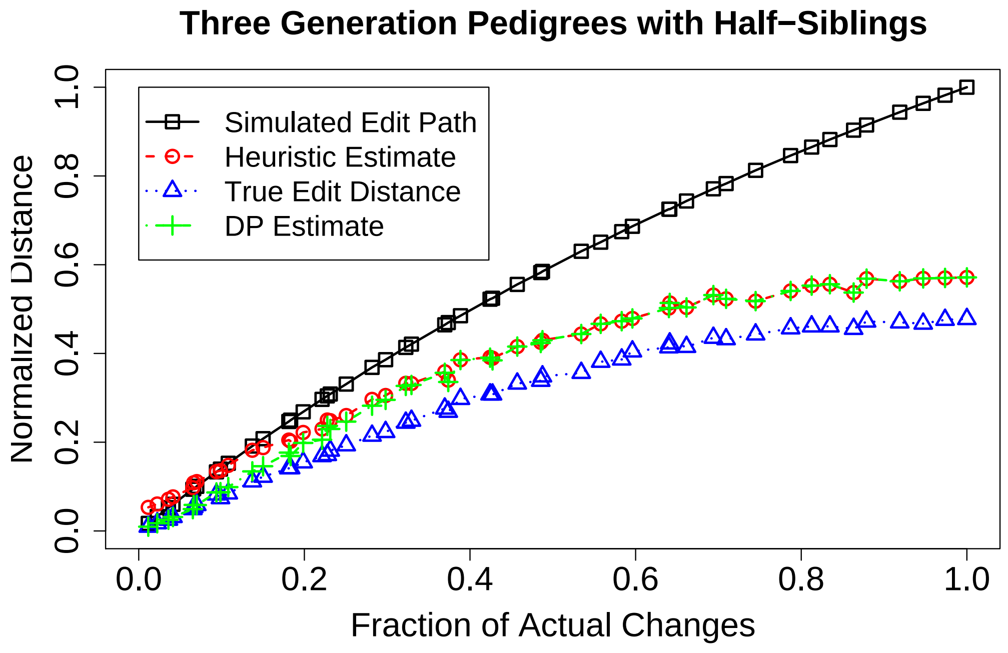
We evaluated the three most general algorithms on random pairs of pedigrees at a range of distances. For each pair, the first pedigree graph was drawn from a Wright-Fisher simulation where every generation has a fixed number of individuals, n, and each monogamous parent-pair has a number of offspring drawn from the Poisson distribution with mean λ. The second pedigree was chosen based on the first in one of two ways. To generate a monogamous second pedigree, a proportion x of non-founders chose a couple uniformly at random to be their parents. To generate a non-monogamous second pedigree, a proportion x of non-founders chose a parent of each gender uniformly and independently at random. This proportion x, called the fraction of actual changes, takes values in the interval $[0, 1]$. In both cases, only the most recent generation was labeled, to simulate the case where there is genetic data available only for the most recent generation. Both simulated pedigrees contain no inter-generational mating events.

We compared three different normalized distance estimates for pedigrees P and Q.

1. Simulated Edit Path Length: $A_{P,Q}/(|E(P)| + |E(Q)|)$
2. Random-Matching Heuristic Estimate: $\hat{D}_{P,Q}/(|E(P)| + |E(Q)|)$
3. True Edit Distance: $D^*_{P,Q}/(|E(P)| + |E(Q)|)$
4. DP Estimate: $\tilde{D}_{P,Q}/(|E(P)| + |E(Q)|)$

where $A_{P,Q}$ is the number actual edge changes in the simulation (often larger than the true edit distance because the edit path taken in the simulation was longer than the shortest edit path), $\hat{D}_{P,Q}$ is edit distance as computed by the random matching heuristic, $D^*_{P,Q}$ is the true edit distance, computed by the branch-and-bound algorithm, and $\tilde{D}_{P,Q}$ is the edit distance as calculated by the DP algorithm, modified so that it returns the random-matching estimate when it does not find a matching within the required distance threshold, set here to $d = 8$.

These distances are plotted in Figure 5.2 against the fraction of pedigree edges changed x during simulation. Figure 5.2 also shows the difference between the random-matching and true edit distances. (The monogamous pedigree results are not shown due to their similarity to the half-sibling case.) The simulations were performed on small three generation pedigrees, so that the edit distance could be computed using the branch-and-bound algorithm, which has exponential running time. From the simulations, it is easy to conclude that both the heuristic algorithm and the DP algorithm provide reasonable estimates of the edit distance. This is particularly true for a small fraction of actual changes, i.e. when the two pedigrees being compared are very similar to each other.



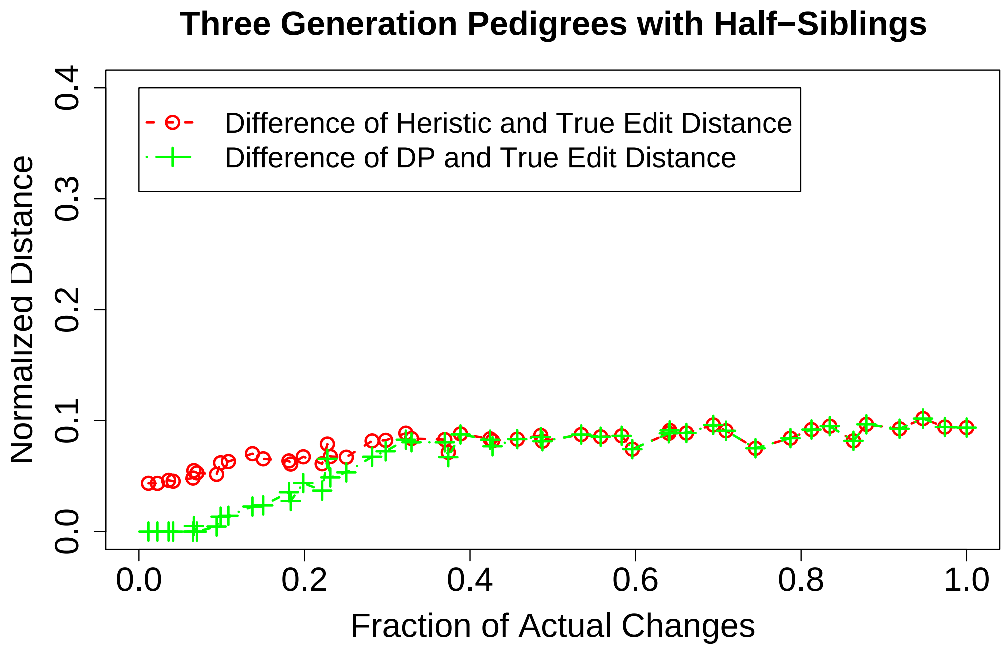


Figure 5.2: **Comparing Different Distances Estimates.** With $n = 14$ and $\lambda = 3$, there were 2500 pairs of pedigrees simulated. Each point is an average of 50 simulations. The values of n and λ were chosen such that the branch-and-bound algorithm would finish computing the true edit distance. The random matching heuristic yields an estimated edit distance which is fairly close to the true edit distance. The DP algorithm performs nearly perfectly for small numbers of actual changes, while it returns the solution found by the random-matching heuristic when it cannot find a solution for parameter $k = 8$. The upper panel shows the accuracies of each algorithm. The lower panel shows the difference in accuracy between the true edit distance and each distances returned by the random-matching heuristic and DP algorithm.

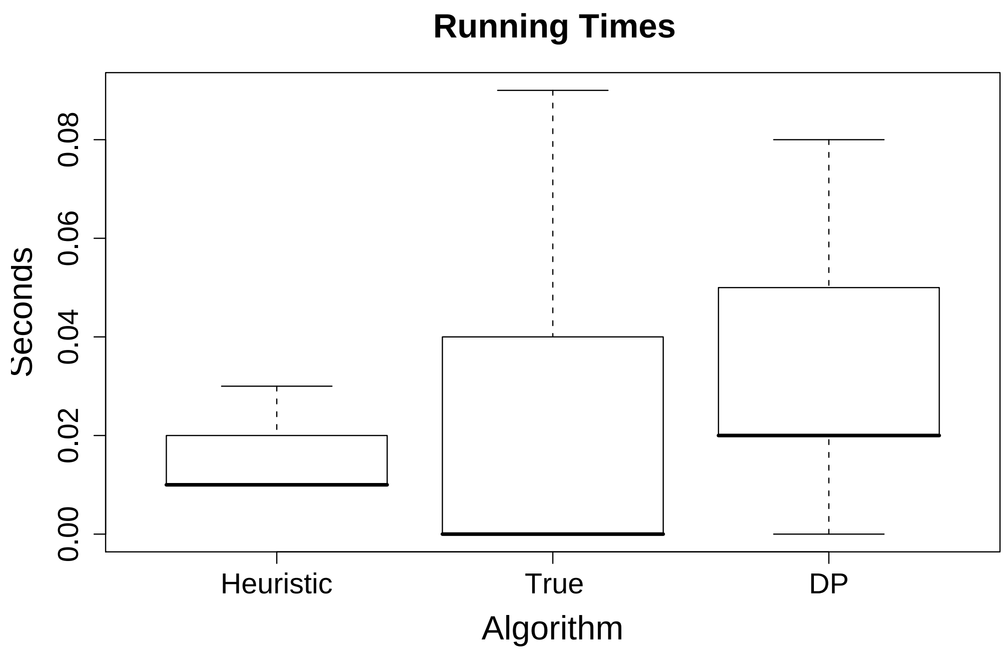


Figure 5.3: **Running Times.** These are box plots comparing the running times of the three different algorithms: heuristic algorithm, branch-and-bound algorithm, and the DP algorithm. The heavy line is the median, the rectangle indicates the first and third quartiles. In this case the median is coincident with the first quartile for all three algorithms. The outliers are not shown; specifically, there are a number of very long execution times for the optimal algorithm.

Figure 5.3 shows the running times for the three algorithms. We see the random-matching heuristic performs reasonably well, both in terms of accuracy and time, so we recommend its use. The DP heuristic agrees with the true edit distance when that distance is small, and agrees with the random-matching estimate otherwise, as expected.

5.4 Two Practical Algorithms for Reconstruction

The work in this section was done as a collaboration between the author, Li, Karp and Halperin [53].

The principal innovation of this method is to reconstruct pedigree graphs *without* reconstructing the ancestral haplotypes. This is the innovation that allows this algorithm to avoid the exponential calculation associated with inferring ancestral haplotypes, and allows the algorithm to be efficient.

The approach we employ is a *generation-by-generation* approach. We reconstruct the pedigree backwards in time, one generation at a time. Of course if we make the correct decisions at each generation, then we will construct the correct pedigree. However, since we use the predictions at previous generations to help us make decisions about how to reconstruct subsequent generations, we can accumulate errors as the algorithm proceeds backwards in time.

Given a set of extant individuals with IBD information available, we want to reconstruct their pedigree. We construct the pedigree recursively, one generation at a time. For example, the first iteration consists of deciding which of the extant individuals are siblings. The next iteration would determine which of the parents are siblings (yielding cousin relationships on the extant individuals).

At each generation, we consider a *compatibility graph* on the individuals at generation g, where the nodes are individuals and the edges are between pairs of individuals that could be siblings. The presence or absence of edges will be determined by a statistical test, discussed later. For the moment, assume that we have such a graph.

Now, we will find sibling sets in the compatibility graph. We do this by partitioning the graph into disjoint sets of vertices with the property that each set in the partition has many edges connecting its vertices while there are few edges connecting vertices from separate sets in the partition. Of course any partitioning method can be used, and later we will introduce a partitioning heuristic. For rhetorical purposes, we will now discuss how to use a Max-Clique algorithm to partition the graph. The graph is partitioned by the following iterative procedure. Iteratively, find the Max-Clique, for all the individuals in the Max-Clique, make them siblings, by creating monogamous parents in generation $g + 1$. Remove those Max-Clique individuals from the graph. Now, we can iterate, by finding the next Max-Clique and again creating a sibling group, etc.

Next, we consider how to create the edges in the compatibility graph. Let individuals k and l be in generation g. Recall that we have an edge in the compatibility graph if k and l could be siblings. To determine this, we look at pairs i and j of descendants of k and l, respectively. Let $\hat{s}_{ij}$ be the observed average length of shared segments between haplotyped individuals i and j. This can be computed directly from the given haplotype data and need only be computed once as a preprocessing step for our algorithm. Now, for a pair of individuals k and l in the oldest reconstructed generation, $X_{i,j}$ is the random variable for the length of a shared region for individuals i, j under the pedigree model that we have constructed so far. Later, we will discuss two models for $X_{i,j}$. For now, consider the test for the edge (k, l)

$$v_{k,l} = \frac{1}{|D(k)||D(l)|} \sum_{i \in D(k)} \sum_{j \in D(l)} \frac{(\hat{s}_{ij} - \mathbb{E}[X_{ij}])^2}{var(X_{ij})} \tag{5.2}$$

where $D(k)$ is the set of extant individuals descended from ancestor k, and $D(k)$ is known based on the pedigree we have constructed up to this point. We compute $v_{k,l}$, making edges when $v_{k,l} < c$ for all k, l in the oldest generation, g, for some threshold c. Notice that this edge test is similar to a χ^2 test but does not have the χ^2 null distribution, because the term in the sum will not actually be normally distributed. We choose the the threshold, c, empirically by simulating many pedigrees and choosing the threshold which provides the best reconstruction accuracy.

Now, we need to calculate $\mathbb{E}[X_{i,j}]$ and $Var(X_{i,j})$. We propose two models for the random variable X_{ij}, the outbred model (COP) and the inbred model (CIP). The outbred, COP, model only allows prediction of relationships between two individuals that are unrelated at all previous generations. The inbred model, CIP, allows prediction of a relationship that relates two individuals already related in a previous generation.

5.4.1 IBD Model for Constructing Outbred Pedigrees (COP)

To obtain the edges in the compatibility graph, we do a test for relationship-pairs of the form shown in Figure 5.4. If a pair of extant individuals i and j are related at generation g via a single ancestor at that generation, then the length of the regions they share IBD will be distributed according to the sum of two exponential variables, specifically, $exp(2(g-1)\lambda)$. This is the waiting time, where time corresponds to genome length, for a random walk to leave the state of IBD sharing. So, we have $X_{ij} = X_1 + X_2$ where $X_i \sim exp(2(g-1)\lambda)$. We must consider the sum of the two exponential random variables, because X_i is the length of the IBD region conditioned on starting at an IBD position. Therefore from an arbitrary IBD position, we need to consider the length of the IBD region before arriving at that position, X_1, and the length after that position, X_2.

Due to these random variables being exponentially distributed, we can quickly analytically compute $\mathbb{E}[X_{ij}]$ and $Var(X_{ij})$. Of course, the edges created respect the outbreeding constraint, such that a pair of individuals, k and l at the gth generation can only have an edge between them in the compatibility graph if none of the extant individuals in $D(k)$ and $D(l)$ are related to each other at a previous generation.

5.4.2 IBD Model for Constructing Inbred Pedigrees (CIP)

We will do a random-walk simulation to allow for inbreeding, resulting in an algorithm with exponential running-time. The number of states in the IBD process is exponential

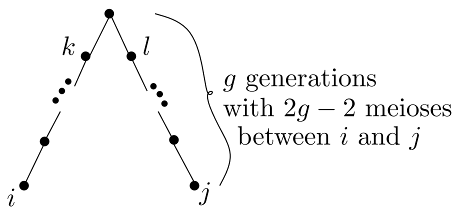


Figure 5.4: **Pair of Individual Related at Generation g.** To test whether individuals k and l are siblings at generation g, we look at the distribution on the length of genetic regions shared IBD between all pairs of i and j descended from k and l, respectively.

in the number of meioses in the graph relating individuals i and j. So, the random-walk simulation is exponential in the size of the inferred pedigree.

For individuals k and l in generation g, and their respective descendants i and j, we consider the case given in Figure 5.5. The triangles represent the inferred sub-pedigree containing all the descendants of the individual at the point of the triangle, and individuals at the base of the triangle are extant individuals. Note that the triangles may overlap, indicating shared ancestry at an earlier generation (i.e. inbreeding).

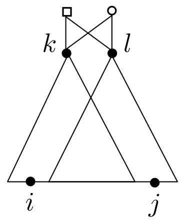


Figure 5.5: **Test Case.** Specific individuals in the pedigree are indicated with either circles or squares. The triangle represents all the descendants of a particular individual. This represents the case where individuals i and j are cousins via the oldest generation.

Brief Description of the IBD Simulation. Let $X_{i,j}$ be the length of a shared region based on the pedigree structure of the model. In order to estimate this quantity, we can sample random walks in the space of inheritance possibilities. Specifically, consider the inheritance of alleles at a single position in the genome. When there are n non-founder individuals, define an inheritance vector as a vector containing $2n$ bits, where each pair of bits, $2i$ and $2i+1$, represents the grand-parental origin of individual i's two alleles. Specifically, bit $2i$ represents the maternal allele and is zero if the grand-paternal allele was inherited and is one otherwise. Similarly, bit $2i+1$ represents the paternal allele of individual i. The set of possible inheritance vectors comprise the 2^{2n} vertices of a $2n$-dimensional hypercube, where n is the number of non-founders in the pedigree. A random walk on the hypercube represents the recombination process by choosing the inheritance vectors of neighboring regions of the genome.

Given an inheritance vector, we can model the length, in number of positions, of the genomic region that is inherited according to that inheritance vector. The end of that genomic region is marked by a recombination in some individual, and constitutes a change in the inheritance vector. The random walk on the hypercube models the random recombinations, while the length of genomic regions are modeled using an exponential distribution. This model is the standard Poisson model for recombinations. Details can be found below.

Poisson Process. Given a pedigree and individuals of interest i and j, we will compute the distribution on the length of shared regions. Here we mean sharing to be a contiguous region of the genome for which i and j have at least one IBD allele at each site.

We can model the creation of a single zygote (i.e. haplotype) as a Poisson process along the genome where the waiting time to the next recombination event is exponentially distributed with intensity $\lambda = -ln(1-\theta)$ where θ is the probability of recombination per meiosis (i.e. per generation, per chromosome) between a pair of neighboring loci. For example, if we think of the genome as being composed of 3000 blocks with each block being 1Mb in length and the recombination rate $\theta = 0.01$ between each pair of neighboring blocks, then we would expect 30 recombinations per meiosis, and the corresponding intensity for the Poisson process is $\lambda = 0.01$.

Now, we have $2n$ meioses in the pedigree, with each meiosis creating a zygote, where n is the number of non-founder individuals. Notice that at a single position in the genome, each child has two haplotypes, and each haplotype chooses one of the two parental alleles to copy. These choices are represented in an inheritance vector, a binary vector with $2n$ entries. The 2^{2n} possible inheritance vectors are the vertices of a $2n$-dimensional hypercube. We can model the recombination process as a random walk on the hypercube with a step occurring each time there is a recombination event. The waiting time to the next step is drawn from $exp(2n\lambda)$, the meiosis is drawn uniformly from the $2n$ possible meioses, and a step taken in the dimension that represents the chosen meiosis. The equilibrium distribution of this random walk is uniform over all the 2^{2n} vertices of the hypercube.

Detailed IBD Simulation. Recall that we are interested in the distribution of the length of a region that is IBD. Recall that IBD is defined as the event that a pair of alleles are inherited from the same founder allele. For individuals i and j, let D be the set of hypercube vertices that result in i and j sharing at least one allele IBD. Given x_0 a hypercube vertex drawn uniformly at random from D, we can compute the hitting time to the first non-IBD vertex by considering the random walk restricted to $D \cup \{d\}$ where d is an aggregate state of all the non-IBD vertices. The hitting time to d is the quantity of interest. In addition, we also need to consider the length of the shared region before reaching x_0, which is the time reversed version of the same process, for the same reason that we summed two exponential random variables for the outbred model in Section 5.4.1.

The transition matrix for this IBD process is easily obtained as $Pr[x_{i+1} = u | x_i = v] = \frac{1}{2n}$ when vertices u and v differ by exactly one coordinate, and $Pr[x_{i+1} = u | x_i = v] = 0$ otherwise. Transitions to state d are computed as $Pr[x_{i+1} = d | x_i = u] = 1 - \sum_{v \in D} Pr[x_{i+1} = v | x_i = u]$.

Now we can either analytically compute the hitting time distribution or estimate the distribution by simulating paths of this random walk. Since the number of IBD states may be exponential, it may be computationally infeasible to find eigenvectors and eigenvalues of the transition matrix [24]. We choose to simulate this random walk and estimate the distribution. This simulation is at worst exponential in the number of individuals.

5.4.3 Heuristic Graph Partitioning Method

The Max-Clique algorithm was only used to illustrate the graph partitioning method. For both the COP and CIP algorithms we use an efficient heuristic for partitioning the vertices of the *compatibility* graph. This method is beneficial, because it looks for densely connected sets of vertices, rather than cliques, which allows for missing edges.

The algorithm is used to partition the vertices, $V(G^g)$, of graph G^g, into a partition $P = \{P_1, P_2, ..., P_C\}$, where $P_i \cap P_j = \emptyset$ for all i, j, and $V(G^g) = \cup_{i=1}^{C} P_i$. For a given partition set, let E_i be the edges of the subgraph induced by vertices P_i. We wish to find a partition such that each set in the partition is a clique or quasi-clique of vertices. The objective function is to find a partition that maximizes $\sum_{i=1}^{C}(a+1)|E_i| - \binom{|P_i|}{2}$ where $a = 0.1$ is a parameter of the algorithm. This objective function is chosen, because it is equivalent to $\sum_{i=1}^{C} a|E_i| - \left(\binom{|P_i|}{2} - |E_i|\right)$, where the term in parentheses is the number of missing edges in the clique. Details of the partitioning method can be found in Karp and Li [45].

The running-time of this graph-partitioning heuristic largely determines the running-time of the pedigree reconstruction algorithm. The partitioning algorithm runs in polynomial time in the size of the graph, if the size of each set in the partition is constant. The step of creating the graph is polynomial in the size of the previous generation graph. Clearly it is possible, if no relationships are found, for the size of the graph at each generation to double. So, in the worst case, this algorithm is exponential. However, in practice this method performs quite quickly for constructing eight-generation pedigrees on large inputs.

5.4.4 Simulation Results

Pedigrees were simulated using a variant of the Wright-Fisher model with monogamy. The model has parameters for a fixed population size, n, a Poisson number of offspring λ, and a number of generations g. In each generation g, the set of n_g individuals is partitioned into $n_g/2$ pairs, and for each pair we randomly decide on a number of offspring using the Poisson distribution with expectation $\lambda = 3$.

The human genome was simulated as 3,000 regions, each of length 1Mb, with recombination rate 0.01 between each region and where each founder haplotype had a unique allele for each region. The assumption here is that IBD information can be given as input to our method. This is not highly restrictive, since if two individuals have some phasing of their genotypes for which there is a common haplotype for a 1Mb region (typically 500 SNPs), they are likely IBD. Notice that Stankovich et al. require haplotypes as input to their method [87], which can be thought of as a form of IBD input.

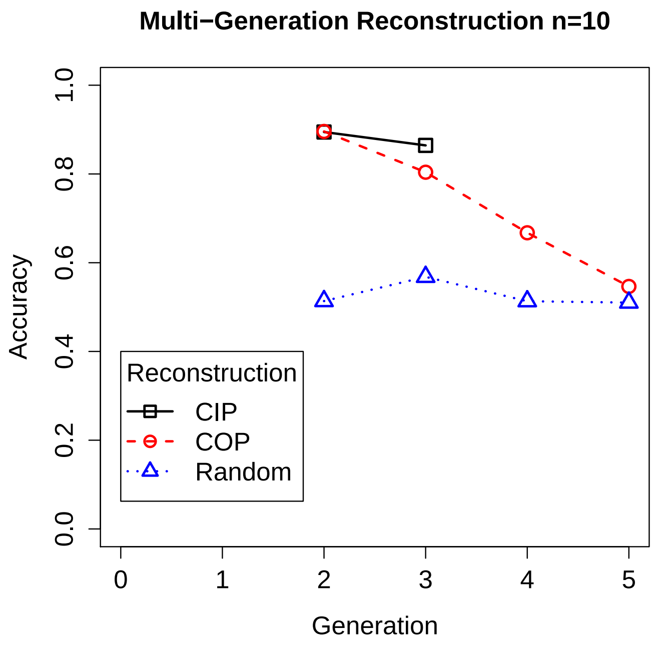


Figure 5.6: **Reconstruction under High Inbreeding.** Here the pedigrees were simulated with a fixed population size of $n = 10$ individuals per generation. Over multiple generations, this results in a high level of inbreeding. The inaccuracy on the y-axis is measured by computing the kinship distance. (Reconstruction accuracy of 50 simulated pedigrees were averaged.)

In each experiment we end up having the true pedigree P generated by the simulation, as well as an estimated pedigree Q. We evaluate the accuracy of the estimated pedigree by using the random-matching heuristic estimate of the edit distance from the previous section. Specifically, if $\hat{D}_{P,Q}$ is edit distance as computed by the random matching heuristic, the normalized edit accuracy is

$$1 - \hat{D}_{P,Q}/(|E(P)| + |E(Q)|).$$

This is the 'accuracy' quantity plotted on the y-axis of the plots.

Selecting Parameters. Notice that there is some interaction between setting threshold c for creating edges in the compatibility graph and the parameter a for how much the quasi-cliques can differ from actual cliques. For a fixed choice of parameter a, we simulated pedigrees and reconstructed them in order to choose the threshold c that gave the best performance. There is competition between how much the quasi-cliques differ from cliques, i.e. how large a is, and the permissiveness of the edge-creation threshold. The larger a is the fewer edges must be created and the smaller c must be in order to maintain accuracy.

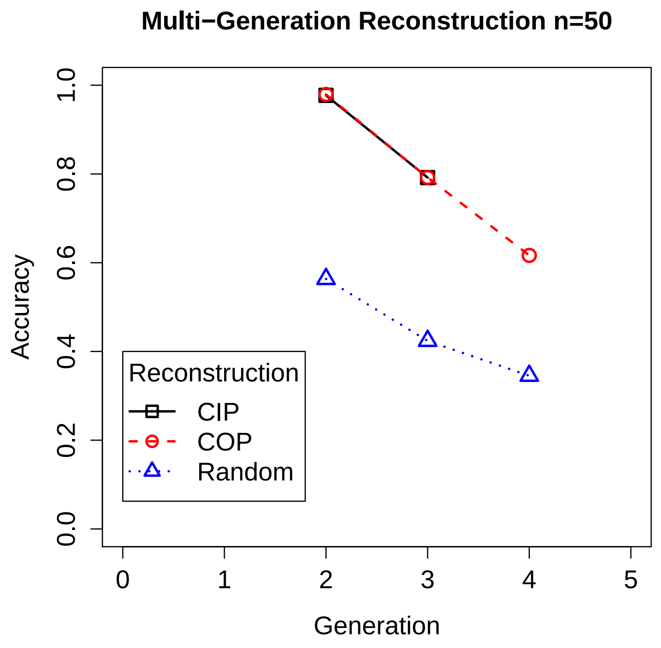


Figure 5.7: **Reconstruction under Less Inbreeding.** Pedigrees here were simulated with a population size of $n = 50$. The y-axis show inaccuracy measured by kinship distance. (Reconstruction accuracy of 50 simulated pedigrees were averaged.)

(Data not shown.) However, for both algorithms we find that $a = 0.01$ and $c = 0.7$ yield the best performance.

Accuracy of COP versus CIP. We compare the COP and CIP methods on inbred pedigree simulations with high and moderate inbreeding, respectively $n = 10$ and $n = 50$, in Figures 5.6 and 5.7. These figures show the kinship-based inaccuracy on the y-axis and the number of generations in the reconstructed pedigree on the x-axis. As the depth of the estimated pedigree increases the error in the kinship of the estimated pedigree increases. However the accuracy is still much better than the accuracy of a randomly constructed pedigree, which is the highest, i.e. worst, line in each figure. CIP performs better on more inbred populations, which we would expect from the modeling assumptions. The running time on the 50 replicates of the $n = 50$ pedigree was 455.32s for COP and 1818.56s for CIP as a total running-time for all the simulated generation sizes.

Size of Reconstructed Pedigrees. Both the COP and CIP methods can reconstruct pedigree with four generations. The COP method for outbred pedigrees can reconstruct pedigrees going back to the most-recent common ancestor of the extant individuals. Provided with enough individuals, the method can construct pedigrees many generation deep. For example, given 400 individuals the method can construct 6 generations. As Figure 5.8

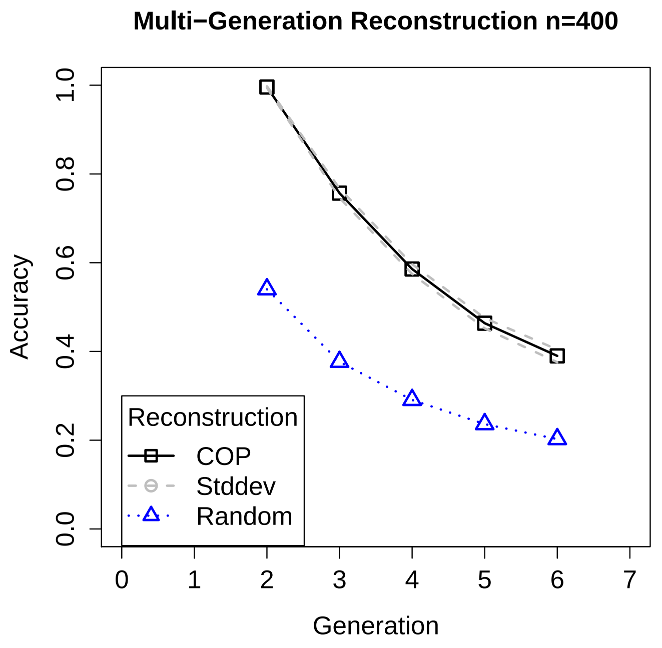


Figure 5.8: **Reconstruction for Deep Pedigrees.** Pedigrees here were simulated with a population size of $n = 400$. (Reconstruction accuracy of 50 simulated pedigrees were averaged.)

shows, the performance relative to a random reconstruction method is very good and so is the variance of the COP reconstruction method.

Comparison with GBIRP. We compare our two methods with the state-of-the-art method, called GBIRP, by Stankovich et al. [87]. Since GBIRP is limited to small pedigrees, we compare the methods on three-generation simulated pedigrees with population size $n = 10$. The simulated pedigrees are connected graphs, so we can look at two accuracy measures, relationships that are mis-specified and relationships that should have been predicted but where not. GBIRP predicts meiosis distance, g_{ij}, between pairs of individuals, i, j, without inferring pedigree relationships. In order to compare GBIRP with the actual pedigree, we extract the minimum number of meiosis, a_{ij}, separating every pair of individuals i and j in the simulated pedigree. From our predicted pedigrees, we again extract a minimum meiosis distance $p_{i,j}$. Now can compute L_1 distances between the actual and predicted meiosis distances. These quantities are $\sum_{i<j:g_{i,j}\neq\infty} |a_{i,j} - g_{i,j}|$, and $\sum_{i<j:p_{i,j}\neq\infty} |a_{i,j} - p_{i,j}|$. This is the number of meioses, or edges in the pedigree graph, which are wrong on paths connecting all pairs of extant individuals. This is plotted in the left panels of Figures 5.9 and 5.10. Now, for a pair of extant individuals, there is always some relationship in the simulated pedigree, since it is a connected graph. But it is possible that one of the inference algorithms did not predict a relationship. Specifically this quantity is $\sum_{i<j:g_{i,j}=\infty} 1$, and $\sum_{i<j:p_{i,j}=\infty} 1$, and it is

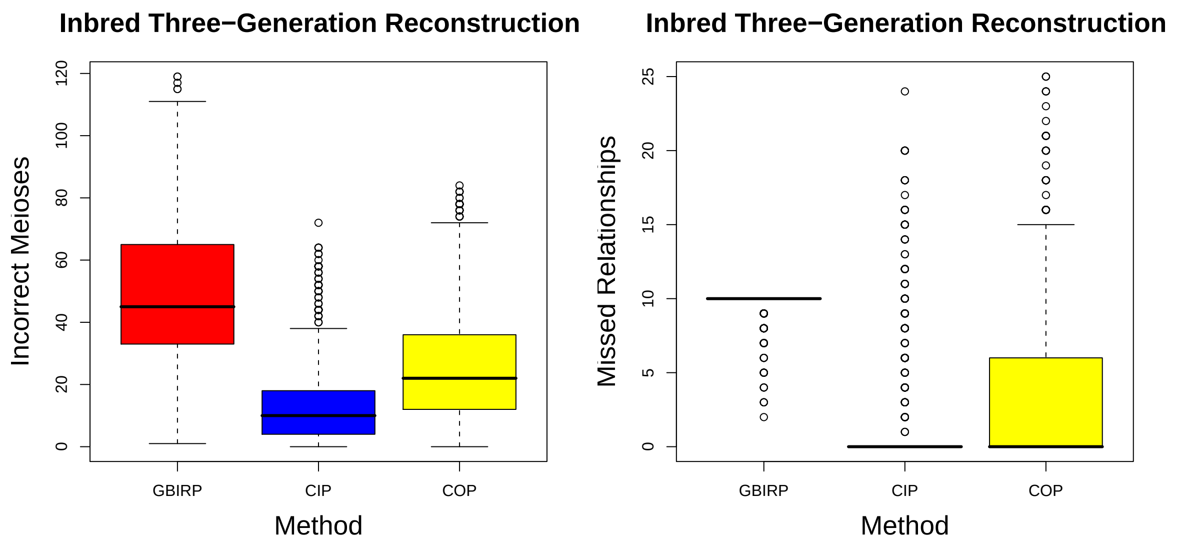


Figure 5.9: **Comparison with GBIRP on Inbred Simulations.** The three-generation pedigrees here were simulated with $n = 10$ extant individuals, since GBIRP could not process larger pedigrees. The accuracy of 1000 simulated pedigrees were computed and plotted. Here the CIP method performs the best, i.e. closest to zero on both plots.

plotted in the right panel of both figures.

Figure 5.9 was done with the simulation method described above. However, in Figure 5.10, to obtain pedigrees with even more outbreeding, a large population size was simulated and a connected sub-pedigree with the desired number of extant individuals was extracted from the large simulation. Notice that with more inbred pedigrees, under this measure of accuracy, the CIP algorithm performs superior to both the COP and the GBIRP methods. The accuracy of COP and CIP increase on the inbred data as compared to the outbred data, perhaps because inbreeding increases the apparent IBD making relationships easier to detect.

Relationships in the HapMap and Wellcome Trust Data. A recent paper by Pemberton et al. [75] reported many familial relationships among MKK individuals in HapMap and few relationships among the CEU and YRI individuals. The method they used did not reconstruct pedigrees, but estimated pair-wise relationships. As a follow-up to their study, we ran our method on the parents of the CEU and YRI trios (for which Pemberton et al. found no relationships) and on the unrelated MKK individuals (for which Pemberton, et al. found 9 first degree relationships). Our results contradicted theirs in that we found no evidence of first degree relationships among the MKK individuals and evidence of 2nd and 4rd cousin relationships in the YRI and CEU, respectively. We also ran our method on the Wellcome Trust individuals having at least 85% identity by state (IBS) and found that some individuals look like 2nd cousins.

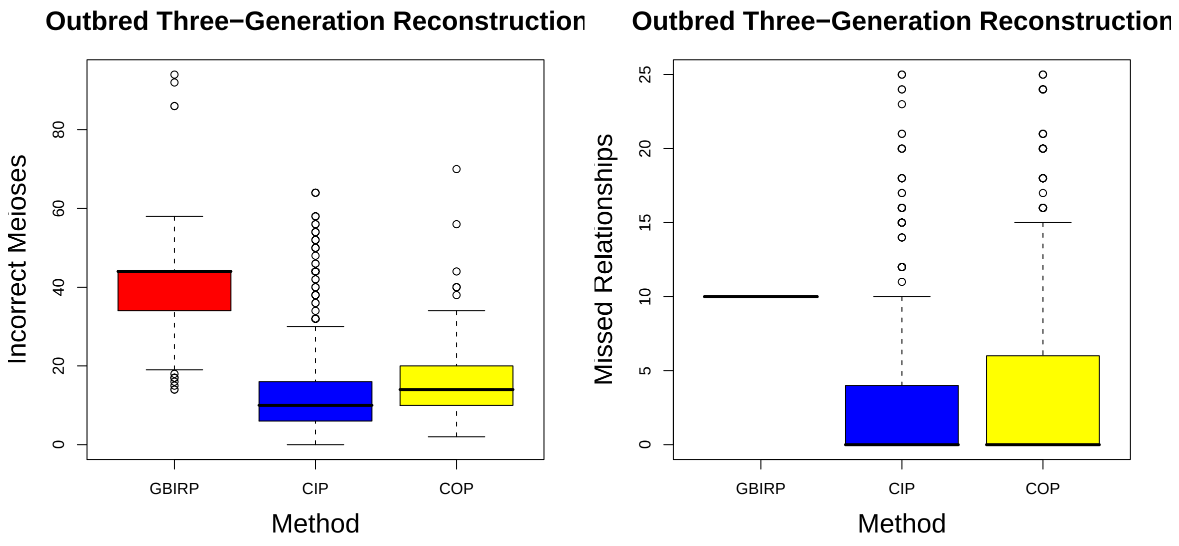


Figure 5.10: **Comparison with GBIRP on Outbred Simulations.** The three-generation pedigrees here were simulated with $n = 10$ extant individuals, since GBIRP could not process larger pedigrees. Here, the simulated pedigree relating the extant individuals was outbred. The accuracy of 1000 simulated pedigrees were computed and plotted. All methods perform better than they did on the inbred data set. Over all, the COP method performs best on the outbred data.

Taking the data from the individuals of interest, between every pair of people we inferred IBD states (0,1, or 2 alleles shared IBD) along the genome and gave those predictions as input to our reconstruction method. To infer IBD, we used a method applied to consecutive, non-overlapping 1Mb windows of the genome: if the two individuals are homozygous for the same alleles across the window, then the IBD state is two shared alleles; if the two individuals have some phasing of the window such that one haplotype can be shared in the region, then the IBD state is one shared allele. Note, that since our reconstruction method takes the IBD predictions as input, a more sophisticated method may be used, such as the HMM used by Plink [78] or the hashing method used by GERMLINE [34]. However, we believe that this simple method is sufficient, because it is unlikely for a pair of non-IBD individuals to share a haplotype for a whole 1Mb region.

Our reconstruction method infers the average length of shared regions between every pair of individuals from the input IBD states. For fixed IBD states, there are multiple sets of shared segments that can explain the IBD states. However, if we assume that segments can only begin and end at transitions from one IBD state to another, then the number of shared segments is fixed. Since the sum of the lengths of the shared segments is also fixed, the expected length of the shared segments is the same regardless of the particular explanation chosen. The variance is not the same, but the edge test only depends on the expectation. Therefore, the estimation of average length of shared regions from the IBD states is straightforward.

For the MKK, CEU, and YRI HapMap individuals, we ran our COP reconstruction method. For the MKK unrelated individuals, we found some individuals related who are 3rd cousins and related by a 5th generation ancestor. For the CEU individuals, we found some individuals related by a 6th generation ancestor, meaning they are 4th cousins. For the YRI individuals, we found 2nd cousins. These results are not consistent with the results found by Pemberton et al [75]. This can be explained by possible errors in the inferences made by the method of Pemberton et al., by our method, or both. We found that some of the first-degree relatives predicted by Pemberton et al. in the MKK individuals did not pass close inspection of the data. For example, true parent-child pairs must share a whole chromosome by Mendelian inheritance, since the child inherits a chromosome from the parent. This sharing happens regardless of the transmitted chromosome being recombinant. Several parent-child pairs predicted by Pemberton et al. had many 1Mb regions in disagreement, and had 30 disagreeing SNPs out of 500 SNPs in a typical window. Furthermore there is a set of three individuals, two pairs of which they predicted to be full siblings, yet the third pair of individuals was not predicted to be siblings. Since full sibling relationships must be transitive, there is clearly an error in their prediction.

Taking the individuals from the Wellcome Trust data that have at least 85% identity-by-state (IBS) with some other individual, we ran our IBD inference method on the genotypes and ran the COP reconstruction method on the IBD inferences. We found some 2nd cousins, meaning individuals related via some 4th generation ancestor.

For all these results, it should be noted that every relationship prediction method has difficulty making reliable predictions. Our method is heavily dependent on accurate IBD predictions and can be misled by genotyping errors. Such errors lead our method to under-predict rather than over-predict relationships, since our simple determination of IBD is disrupted by a single dis-agreeing SNP. Indeed, it is important not to phase the genotypes before predicting IBD, since the phasing process can lead to incorrectly imputed missing genotypes and disrupted IBD estimates. It is quite possible that all relationship prediction methods are very sensitive to genotyping errors. Due to these difficulties, we believe that these aspects of relationship prediction should continue to be investigated. Chapter 6 proposes potential future work relating both to IBD inference and pedigree reconstruction.

5.5 Discussion

In this chapter, we introduce a novel formulation for pedigree graphs based on the idea of d-splits. We also introduce the problem of comparing pedigree graphs via an edge cut-and-paste distance, and we show this problem to be APX-hard. We give several exact algorithms and several heuristics for this problem. The heuristics were implemented and compared with the optimal edit distance on simulated examples.

One interesting open problem is exploring an alternative definition of edit distance based on fractional matchings. Instead of minimizing over all one-to-one matchings of nodes in the two pedigrees, we could allow a node of P to be matched to multiple nodes in Q with weights summing to one. Such a distance could be easier to compute, although the biological interpretation is less clear. It would also be interesting to explore the relationship between the definition presented in this chapter, and this alternate definition.

Another open problem of interest is how these algorithms work on non-regular pedigrees, i.e. with inter-generational mating. The simulations used here were based on the Wright-Fisher model and did not allow any inter-generational mating events. It may be of interest to model the pedigrees using a birth-death model such as the Moran model where inter-generational mating is allowed. Such a simulation would allow the evaluation of the distance heuristics on non-regular pedigrees.

Also of great interest is understanding pedigrees having missing genealogical information, where not all the parent-child edges or the descendant splits are known in one or both of the pedigrees. There are two issues of interest. First, how robust is the descendant-split formulation to missing information? Second, is there an edit distance between pedigrees that can allow missing information?

Algorithms for comparing pedigrees are useful to researchers who currently build pedigrees manually. In particular it is quite possible to have multiple genealogical sources that might produce different pedigrees. The ability to compare these pedigrees to find the differences is essential. This chapter presents one such approach.

We note that our methods for pedigree reconstruction are limited to a restricted scenario in which there is monogamy and the generations are synchronous. If monogamy is broken then our approach will not work since the sibling relationships in the compatibility graph at each level will not be a simple partition. It is plausible that a different graph formulation may still provide an accurate solution to more complex pedigrees, however the exact formulation that will resolve such pedigrees is currently unknown and is left as an open challenge.

There are significant open challenges with pedigree reconstruction. For example, it would be nice to obtain confidence values on the inferred pedigree edges. However this seems very difficult, even if we can draw pedigrees from the posterior distribution of pedigree structures given the data. Since edges in a pedigree are not labeled, obtaining confidence values for a pedigree P would translate to: drawing pedigree samples, Q, from the distribution, identifying the edges in P and Q that provide the same relationships, and scoring the edges of P according to the probability of pedigree Q. As discussed in Section 5.3.2, the second step, identifying the edges in P and Q that provide the same relationships, is a hard problem.

Chapter 6

Conclusions

6.1 Progress on Motivating Questions

The work in this thesis is aimed at a number of important problems for both population and family genetics. Issues of privacy, accuracy of association studies, estimates of recombination breakpoints, and pedigree reconstruction have all been considered. The methods used to approach these problems have been both statistical and computational.

In Chapter 2, we discussed models for population genetic data. Two problems were introduced and solved. The first problem involved identifying individuals in the pooled data of a genetic study. This has applications to the privacy of genetic data and underlines the result of Sankararaman, et al. [81] that only pooled data from a small number of unlinked loci can be released publicly without violating the privacy of the study participants. The second problem was to efficiently enumerate the perfect phylogenies compatible with binary partial characters. The general version of this problem is NP-hard, however, when the data satisfy the *rich data hypothesis* the enumeration can be done efficiently. This has applications to computing probabilities of data with missing values under the coalescent with the infinite sites mutation model.

In Chapter 3, pedigree models for families of related individuals were introduced. Identity by descent was introduced and used to compute inheritance probabilities. The Lander-Green and Elston-Stewart algorithms were discussed, and the hardness of the pedigree likelihood calculation given haplotype data was established. The hardness of this calculation is important due to the potential of next-generation sequencing methods to produce haplotype data.

Inference under the pedigree model is challenging. In Chapter 4, three new algorithms were introduced for inferring various quantities of interest. The Gibbs sampling algorithm for a small number of sites was introduced to infer haplotypes for all the individuals in a large pedigree. This algorithm is essentially a variant of the Elston-Stewart algorithm and has applications to improving the accuracy of association studies on pedigree data sets. For inferring recombination breakpoints from haplotype data, we discussed an HMM algorithm that is a variant of the Lander-Green algorithm and implements the forward-backward algorithm for HMMs. This algorithm is important due to the potential for next-generation sequencing methods to produce haplotype data. The last algorithm introduced was a method to reduce the number of IBD hidden state for the pedigree HMM. For certain pedigrees, an exponential algorithm can reduce the state space exponentially resulting in HMM algorithms with faster running times. Methods, such as this, for improving the efficiency of the pedigree likelihood calculation are important due to the many and varied uses for pedigree likelihoods, including linkage analysis, association testing, and IBD inferences.

In Chapter 5, three pedigree reconstruction algorithms were introduced along with several algorithms for evaluating the accuracy of reconstruction estimates. The first pedigree reconstruction algorithm was of theoretical interest. The second and third reconstruction algorithms were of practical utility for inferring monogamous, regular pedigrees that relate a set of extant individuals. In order to evaluate the accuracy of a reconstructed pedigree under simulations, it is necessary to compare the simulated pedigree to the reconstructed pedigree. Since only the extant individuals are labeled, somehow the ancestral individuals in the two pedigrees must be mapped onto each other. This mapping yields an edit distance which measures the differences between the two pedigrees. Three algorithms were introduced for computing exact and heuristic edit distances. Simulations showed that the heuristic edit distance algorithm was both fast and reasonably accurate, and the heuristic was used to evaluate the accuracy of the two practical pedigree reconstruction methods.

6.2 Future Problems

This section proposes a number of important and approachable problems. The goal will be to discuss problems that are doable in the near future. These problems involve linkage analysis, sequence data, pedigree reconstruction, combined population-genetic and family models, and IBD estimates.

6.2.1 Pedigree Likelihood Calculations

The pedigree likelihood calculation appears almost everywhere there is an interesting question about pedigrees. Due to the ubiquity of this likelihood, efficient methods for calculating it are incredibly important. We will discuss a few places where this likelihood can be used to solve important problems.

Linkage Analysis

In linkage analysis [95, 72], a position for a hypothetical disease locus is found in the genetic map. Suppose that the positions of all the SNPs are known, $S_1, S_2, ..., S_m$; this is called the genetic map. Now taking the hypothetical disease locus D, try placing it between each pair of SNP markers S_j, S_{j+1}. Infer the recombination rate, θ, between D and the neighboring SNP, S_{j+1}, using the pedigree HMM and the forward backward algorithm. Once the most likely position for D and value for θ has been found, perform the following likelihood ratio (LR) test

$$\frac{L(\theta)}{L(1/2)}$$

Here $L(\theta)$ is the likelihood, or probability of the observations given a particular parameter value θ. In the denominator, with $\theta = 1/2$, the model is for the disease locus to be unlinked. Very high values of the LR test indicate that a position of disease linkage has been found. Experimental work examining the genes in that region of the genome can possibly yield a gene that is related to the production of the disease phenotype.

It is possible that the composite likelihood [97] can be used to more efficiently estimate a reasonable θ. Choose a subset of the data G_k by selecting some individuals or some genotypes. Now approximate the likelihood as a product of the likelihood on each subset of the data

$$\tilde{L}(\theta) = \prod_k L(\theta; G_k).$$

It is clear that this approach would result in faster computation of the likelihood ratio score. The questions, then, are whether this method would yield good accuracy and whether the composite likelihood test is consistent and unbiased.

Sequencing Data

In Chapter 4, estimation of recombination rates from haplotype data was considered. The assumption, there, was that one could obtain haplotypes for the whole chromosome. In practice this is unlikely. From sequencing methods, one would expect to obtain a mixture of haplotype fragments beginning and ending at unexpected positions in the genome.

This leads to a case of mixed haplotypes and genotypes. For this case to be interesting, the ends of haplotypes must occur at different locations in different individuals in the pedigree. Otherwise, the haplotypes that start and end at the same positions in all individuals can easily be converted into multi-allelic genotypes, with an allele for each haplotype. The mixed haplotype-genotype problem is not amenable to the hardness proof techniques used in Chapter 3. However, the haplotype HMM in Chapter 4.2 can easily be revised to handle the mixed case. This is important because the data produced by single-molecule sequencing is more likely to resemble the mixed case than either the haplotype or the genotype cases.

Pedigree Reconstruction

The problem of constructing pedigrees from a set of extant individuals is well motivated, as discussed in Chapter 5. However, in that chapter we considered a somewhat restricted formulation of the pedigree reconstruction problem, i.e. monogomous, regular pedigrees. Furthermore, the practical methods introduced in that chapter have the problem that they may estimate a pedigree on which the probability of the data is zero; this is because the reconstruction method does not compute the likelihood when estimating the pedigree. As explained in that chapter, the likelihood calculation is exponential in the number of individuals in the pedigree and avoiding that calculation is the only way to obtain an algorithm that performs efficiently. However, this leaves open the disturbing possibility of inferring a pedigree on which the data could not possibly be inherited, because Mendelian constraints were not considered. While this is not much of a problem for biallelic data on only extant individuals, it presents a greater problem when trying to generalize the method in Chapter 5 to multi-generational or microsatellite data.

There are several potential research directions pertaining to pedigree reconstruction. Each of the directions suggested here will involve slightly different formulations of the reconstruction problem.

It would be interesting to reconstruct pedigrees in the setting where there is data for all of the individuals in the pedigree. Here, the goal is to find the pedigree graph that maximizes the likelihood of the observed data. In this setting, the challenge would be to correctly predict parent-child pairs in a set of people where all the parent-child pairs are present. While this problem seems trivial at first glance, there is reason to believe that it is both important and non-trivial. This problem formulation is important, because this is the setting in which pedigree reconstruction can be performed most accurately. This problem is non-trivial due to the computational complexity of the likelihood calculation and due to known problems with inferring IBD (see Section 6.2.2 for complete discussion).

Another interesting formulation of the pedigree reconstruction problem is to predict sibling groups in the setting where data is given for the children and not for the parents. Again, the goal would be to find the pedigree graph that maximizes the likelihood of the observed data. Again, due to the computational complexity of the likelihood calculation, this problem is non-trivial. Importantly, this is the setting in which the edit-distance is tractable using a 2-generation matching algorithm. This problem formulation is essentially a statistical version of a similar combinatorial problem introduced for wild populations [83, 6]. The difficulty, here, would be to correctly infer sibling pairs, without access to the parental genotype and still respect Mendelian inheritance.

One approach to both of the suggested problems would be to approximate the likelihood using either variational machine learning approaches [3, 2] or perhaps a composite likelihood [97]. Both approximations would present different potential problems, but both might be more efficient than performing the full likelihood calculation.

Combined Population-Genetic and Pedigree Model

A very interesting direction for long-term work is to combine population-genetic and family-based models in order to compute a joint likelihood of inheritance. Rather than continuing the current research paradigm of doing either a population-genetic study or a family study, the goal is to produce methods that combine the strengths of both models and that can handle mixed data as input. Such a method would model recent population structure using a known pedigree and would model ancient relationships using a coalescent. Here the founder haplotypes of a pedigree would be drawn from a coalescent model and then the haplotypes would be inherited in the pedigree according to the Poisson model for recombinations.

This is a challenging problem, due to computational complexity. However, preliminary investigations, as seen in Chapter 4 and in [51], suggest that such a combined model would be more accurate than either model alone. The PhyloPed algorithm used a combined model to infer haplotypes for short genetic regions. Additionally, the computational issues can be addressed by making certain assumptions, for instance in Chapter 4 and in [48], it was noted that if all the pedigree individuals are haplotyped, the combined model is computationally feasible.

6.2.2 IBD Estimates

From the pedigree reconstruction work in Chapter 5, it became very clear that estimates of IBD are both critical for successful pedigree reconstruction and inaccurate. The reasons for the accuracy problems are two-fold. First, genotyping errors disrupt the IBD predictions of methods that assume fidelity of the data and result in shorter regions of IBD than should be predicted. Second, there is difficulty distinguishing between recent and ancient IBD. Ancient IBD is the result of linkage disequilibrium in the population, but IBD prediction methods desire to predict longer, more recent regions of IBD.

Pedigree IBD Estimates

The most accurate methods for predicting IBD use pedigrees. One can consider the pedigree HMM with the IBD states as hidden states and use the Viterbi algorithm to obtain the most probable path through the hidden states. This most probable path is the IBD estimate at each SNP for all the individuals in the pedigree. However, this calculation is exponential, due to the number of hidden states being exponential in the number of non-founders in the pedigree.

Recent work by Li and Li took advantage of the pedigree structure to infer IBD, but they considered pairs of genotyped individuals, rather than the whole pedigree at once [63]. Their model is essentially an HMM on pairs of individuals with a hidden state for background IBD sharing to account for more ancient IBD. It appears that they need the background IBD state to help distinguish between recent and ancient IBD. They obtain good accuracy with their polynomial-time algorithm.

One potential direction of future work would be to apply variational machine learning methods to this problem. Cluster variational methods have been successfully applied to the linkage analysis problem [3] discussed in Section 6.2.1 and to haplotype inference in pedigrees [2]. It is quite conceivable that variational methods will work as well for IBD inference as they do for haplotype inference in pedigrees.

Another potential direction is to allow for genotyping errors. The pedigree HMM model is easily changed to have error probabilities on emission, meaning that the G_t states in Figure 3.6 would have some probability of misrepresenting the alleles. This change to the model would not change the running-time of the Viterbi calculation, and would result in the algorithm continuing to be exponential in the number of non-founders.

A final modeling change that could result in more accurate predictions is to model recombination interference. It is well known that the distribution of recombinations along the genome does not actually match the Poisson process [65]. In fact, the presence of a recombination at one position suppresses the probability of a nearby recombination, whereas the Poisson process assumes that the relative positions of two recombinations are independent. Finding a way to accurately model interference should improve the accuracy of IBD predictions.

Population IBD Estimates

Population-level IBD estimates appear to be more difficult to predict and have less accuracy than pedigree-based IBD estimates. There are roughly two classes of methods for inferring IBD from populations of individuals, those methods applicable to large data sets, such as GERMLINE [34] and fastIBD [11], and those methods having higher accuracy but applied to smaller data sets, such as Plink [78]. Some of the former methods use hashing while the latter methods often rely on HMMs to make IBD predictions.

One very obvious direction is to extend population-based IBD prediction methods to allow for genotyping errors. This is very important because a single genotyping error can interrupt an IBD region and result in shorter IBD segments than should have been predicted. The population-based HMMs for IBD can be modified in a similar manner to that suggested above for the pedigree IBD HMM. The emission probability can be changed to have some probability of emitting an erroneous allele. The result of modeling genotyping errors should be more accurate IBD predictions.

Another direction is to consider IBD inference from sequencing data. Because of the resolution of genetic variation that can be observed with sequencing data, this requires also modeling the possibility of novel mutations that might interrupt IBD regions. Mutations would have a similar effect on IBD prediction accuracy as genotyping errors in that they would interrupt the prediction of a longer IBD region. However, novel mutations might actually appear in multiple individuals, so their signature in the data would match a phylogeny, rather than occurring independently in each individual as genotyping errors do.

One could also attempt to more accurately model IBD for inbred populations. When there is inbreeding, it is no longer sufficient to describe the IBD states as 0, 1, and 2 alleles shared. Instead, it is possible for two individuals to share all four alleles IBD. The condensed identity states introduced by Jacquard [42] are the complete IBD classes for a pair of inbred individuals. It would be interesting to create an HMM for a pair of inbred individuals that infers the IBD state as one of the condensed identity states. The main question is whether this additional model complexity produces more accurate IBD inferences than the simple 0,1,2-allele IBD HMM.

Bibliography

[1] GR Abecasis, SS Cherny, WO Cookson, and LR Cardon. Merlin-rapid analysis of dense genetic maps using sparse gene flow trees. *Nature Genetics*, 30:97–101, 2002.

[2] C. A. Albers, T. Heskes, and H. J. Kappen. Haplotype inference in general pedigrees using the cluster variation method. *Genetics*, 177(2):1101–1116, October 2007.

[3] Cornelis A. Albers, Martijn A. R. Leisink, and Hilbert J. Kappen. The cluster variation method for efficient linkage analysis on extended pedigrees. *BMC Bioinformatics*, 7(S-1), 2006.

[4] K. G. Ardlie, L. Kruglyak, and M. Seielstad. Patterns of linkage disequilibrium in the human genome. *Nature Reviews Genetics*, 3, 2002.

[5] J.C. Barrett, S. Hansoul, D.L. Nicolae, J.H. Cho, R.H. Duerr, J.D. Rioux, S.R. Brant, M.S. Silverberg, K.D. Taylor, M.M. Barmada, and et al. Genome-wide association defines more than 30 distinct susceptibility loci for crohn's disease. *Nature Genetics*, 40:955–962, 2008.

[6] T. Y. Berger-Wolf, S. I. Sheikh, B. DasGupta, M. V. Ashley, I. C. Caballero, W. Chaovalitwongse, and S. L. Putrevu. Reconstructing sibling relationships in wild populations. *Bioinformatics*, 23(13):i49–56, 2007.

[7] H. Bickeboller and E. A. Thompson. Distribution of genome shared ibd by half-sibs: Approximation by the poisson clumping heuristic. *Theoretical Population Biology*, 50(1):66 – 90, 1996.

[8] M. Boehnke and N. J. Cox. Accurate inference of relationships in sib-pair linkage studies. *American Journal of Human Genetics*, 61:423–429, 1997.

[9] C. Bourgain, S. Hoffjan, R. Nicolae, D. Newman, L. Steiner, K. Walker, R. Reynolds, C. Ober, and M. S. McPeek. Novel case-control test in a founder population identifies p-selectin as an atopy-susceptibility locus. *American Journal of Human Genetics*, 73(3):612–626, 2003.

[10] D. Brown and T. Berger-Wolf. Discovering kinship through small subsets. *WABI 2010: Proceedings for the 10th Workshop on Algorithms in Bioinformatics*, 2010.

[11] Brian L. Browning and Sharon R. Browning. A fast, powerful method for detecting identity by descent. *The American Journal of Human Genetics*, 88:173–182, 2011.

[12] S. Browning and B.L. Browning. On reducing the statespace of hidden Markov models for the identity by descent process. *Theoretical Population Biology*, 62(1):1–8, 2002.

[13] S. R. Browning, J. D. Briley, L. P. Briley, G. Chandra, J. H. Charnecki, M. G. Ehm, K. A. Johansson, B. J. Jones, A. J. Karter, D. P. Yarnall, and M. J. Wagner. Case-control single-marker and haplotypic association analysis of pedigree data. *Genetic Epidemiology*, 28(2):110–122, 2005.

[14] J.T. Burdick, W. Chen, G.R. Abecasis, and V.G. Cheung. In silico method for inferring genotypes in pedigrees. *Nature Genetics*, 38:1002–1004, 2006.

[15] C. R. Chegireddy and H. W. Hamacher. Algorithms for finding the k-best perfect matchings. *Discrete Applied Mathematics*, 18:155–165, 1987.

[16] W.-M. Chen and G.R. Abecasis. Family-based association tests for genomewide association scans. *American Journal of Human Genetics*, 81:913 – 926, 2007.

[17] X. Chen, L. Liu, Z. Liu, and T. Jiang. On the minimum common integer partition problem. *ACM Trans. on Algorithms*, 5(1), 2008.

[18] Vanessa J. Clark, Susan E. Ptak, Irene Tiemann, Yudong Qian, Graham Coop, Anne C. Stone, Molly Przeworski, Norman Arnheim, and Anna Di Rienzo. Combining sperm typing and linkage disequilibrium analyses reveals differences in selective pressures or recombination rates across human populations. *Genetics*, 175(2):795–804, 2007.

[19] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 2004.

[20] G. Coop, X. Wen, C. Ober, J. K. Pritchard, and M. Przeworski. High-Resolution Mapping of Crossovers Reveals Extensive Variation in Fine-Scale Recombination Patterns Among Humans. *Science*, 319(5868):1395–1398, 2008.

[21] MJ Daly, JD Rioux, SF Schaffner, TJ Hudson, and ES Lander. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29(2):229–32, Oct 2001.

[22] Z. Ding, V. Filkov, and D. Gusfield. A linear-time algorithm for perfect phylogeny haplotyping. *Journal of Computational Biology*, 13(2):522–553, 2006.

[23] D. Doan and P. Evans. Fixed-parameter algorithm for haplotype inferences on general pedigrees with small number of sites. *WABI 2010: Proceedings for the 10th Workshop on Algorithms in Bioinformatics*, 2010.

[24] K. P. Donnelly. The probability that related individuals share some section of genome identical by descent. *Theoretical Population Biology*, 23(1):34 – 63, 1983.

[25] John Durbin. *Modern Algebra: An Introduction.* John Wiley and Sons, Inc., 4th edition, 2000.

[26] J. Eid and et al. Real-Time DNA Sequencing from Single Polymerase Molecules. *Science*, 323(5910):133–138, 2009.

[27] R.C. Elston and J. Stewart. A general model for the analysis of pedigree data. *Human Heredity*, 21:523–542, 1971.

[28] E. Eskin, E. Halperin, and R. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. *Journal of Bioinformatics and Computational Biology*, 1(1):1–20, 2003.

[29] M. Fishelson, N. Dovgolevsky, and D. Geiger. Maximum likelihood haplotyping for general pedigrees. *Human Heredity*, 59:41–60, 2005.

[30] R. A. Fisher. *The Genetical Theory of Natural Selection.* Oxford University Press, USA, 1 edition, April 1930.

[31] I Gallego Romero and C Ober. CFTR mutations and reproductive outcomes in a population isolate. *Human Genet*, 122:583–588, 2008.

[32] D. Geiger, C. Meek, and Y. Wexler. Speeding up HMM algorithms for genetic linkage analysis via chain reductions of the state space. *Bioinformatics*, 25(12):i196, 2009.

[33] Richard E. Green, Johannes Krause, Adrian W. Briggs, Tomislav Maricic, Udo Stenzel, Martin Kircher, Nick Patterson, Heng Li, Weiwei Zhai, Markus Hsi-Yang Fritz, Nancy F. Hansen, Eric Y. Durand, Anna-Sapfo Malaspinas, Jeffrey D. Jensen, Tomas Marques-Bonet, Can Alkan, Kay Prfer, Matthias Meyer, Hernn A. Burbano, Jeffrey M. Good, Rigo Schultz, Ayinuer Aximu-Petri, Anne Butthof, Barbara Hber, Barbara Hffner, Madlen Siegemund, Antje Weihmann, Chad Nusbaum, Eric S. Lander, Carsten Russ, Nathaniel Novod, Jason Affourtit, Michael Egholm, Christine Verna, Pavao Rudan, Dejana Brajkovic, eljko Kucan, Ivan Guic, Vladimir B. Doronichev, Liubov V. Golovanova, Carles Lalueza-Fox, Marco de la Rasilla, Javier Fortea, Antonio Rosas, Ralf W. Schmitz, Philip L. F. Johnson, Evan E. Eichler, Daniel Falush, Ewan Birney, James C. Mullikin, Montgomery Slatkin, Rasmus Nielsen, Janet Kelso, Michael Lachmann, David Reich, and Svante Pbo. A draft sequence of the neandertal genome. *Science*, 328(5979):710–722, 2010.

[34] A. Gusev, J. K. Lowe, M. Stoffel, M. J. Daly, D. Altshuler, J. L. Breslow, J. M. Friedman, and I. Pe'er. Whole population, genomewide mapping of hidden relatedness. *Genome Research*, 19:318–26, 2009.

[35] Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. In *Proceedings of the 6th Annual International Conference on Research in Computational Molecular Biology*, pages 166–175, 2002.

[36] D. Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, 21:12–28, 1991.

[37] Eran Halperin and Richard M. Karp. Perfect phylogeny and haplotype assignment. In *RECOMB '04: Proceedings of the eighth annual international conference on Computational molecular biology*, pages 10–19, New York, NY, USA, 2004. ACM Press.

[38] L. H. Hartwell, L. Hood, M. L. Goldberg, A. E. Reynolds, L. M. Silver, and R. C. Veres. *Genetics: From Genes to Genomes*. McGraw-Hill, New York, 2nd edition, 2004.

[39] Dale Hedges, Dan Burges, Eric Powell, Cherylyn Almonte, Jia Huang, Stuart Young, Benjamin Boese, Mike Schmidt, Margaret A. Pericak-Vance, Eden Martin, Xinmin Zhang, Timothy T. Harkins, and Stephan Zchner. Exome sequencing of a multigenerational human pedigree. *PLoS ONE*, 4(12):e8232, 12 2009.

[40] Jotun Hein, Mikkel H. Schierup, and Carsten Wiuf. *Gene Genealogies, Variation and Evolution : A Primer in Coalescent Theory* . Oxford University Press, USA, February 2005.

[41] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V. Pearson, Dietrich A. Stephan, Stanley F. Nelson, and David W. Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genet*, 4(8):e1000167, 08 2008.

[42] Albert Jacquard. Genetic information given by a relative. *Biometrics*, 28(4):1101–1114, 1972.

[43] Tommi A. Junttila. A note on the computational complexity of a string orbit problem. 2001.

[44] Tommi A. Junttila. New orbit algorithms for data symmetries. *Application of Concurrency to System Design, International Conference on*, 0:175, 2004.

[45] R. M. Karp and S. C. Li. An efficient method for quasi-cliques partition. *Manuscript in preparation*, 2011.

[46] Anderson K.G. How well does paternity confidence match actual paternity? evidence from worldwide nonpaternity rates. *Curr. Anthropol.*, 47:513–520, 2006.

[47] J. F. C. Kingman. The coalescent. *Stoch. Proc. Appl.*, 13:235–248, 1982.

[48] B. Kirkpatrick. Haplotypes versus genotypes on pedigrees. *WABI 2010: Proceedings for the 10th Workshop on Algorithms in Bioinformatics*, 2010.

[49] B. Kirkpatrick. Pedigree reconstruction using identity by descent. *Class project, Prof. Yun Song, 2008. Technical Report No. UCB/EECS-2010-43*, 2010.

[50] B. Kirkpatrick. Haplotypes versus genotypes on pedigrees. *Alg. for Mol. Bio.*, 2011.

[51] B. Kirkpatrick, E. Halperin, and R. M. Karp. Haplotype inference in complex pedigrees. *Journal of Computational Biology*, 17(3):269–280, 2010.

[52] B. Kirkpatrick and K. Kirkpatrick. Pedigree state-space reduction preserving identity by descent. *manuscript in preparation*, 2011.

[53] B. Kirkpatrick, S.C. Li, R. M. Karp, and E. Halperin. Pedigree reconstruction using identity by descent. *RECOMB 2011: Proceedings of the 15th Annual International Conference on Research in Computational Molecular Biology*, 2011.

[54] B. Kirkpatrick, Y. Reshef, H. Finucane, H. Jiang, B. Zhu, and R. M. Karp. Algorithms for comparing pedigree graphs. *CoRR*, abs/1009.0909, 2010.

[55] B. Kirkpatrick, J. Rosa, E. Halperin, and R. M. Karp. Haplotype inference in complex pedigrees. In *RECOMB 2009: Proceedings of the 13th Annual International Conference on Research in Computational Molecular Biology*, pages 108–120, Berlin, Heidelberg, 2009. Springer-Verlag.

[56] B. Kirkpatrick and K. Stevens. Efficiently constructing phylogenies from partial characters. *manuscript in preparation*, 2011.

[57] Bonnie Kirkpatrick. Estimating haplotype frequencies from genotypes of pooled dna. Master's thesis, EECS Department, University of California, Berkeley, Dec 2007.

[58] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning)*. The MIT Press, August 2009.

[59] E.S. Lander and P. Green. Construction of multilocus genetic linkage maps in humans. *Proceedings of the National Academy of Science*, 84(5):2363–2367, 1987.

[60] Adam S. Lauring and Raul Andino. Quasispecies theory and the behavior of rna viruses. *PLoS Pathog*, 6(7):e1001005, 07 2010.

[61] S. L. Lauritzen and N. A. Sheehan. Graphical models for genetic analysis. *Statistical Science*, 18(4):489–514, 2003.

[62] J. Li and T. Jiang. An exact solution for finding minimum recombinant haplotype configurations on pedigrees with missing data by integer linear programming. In *Proceedings of the 7th Annual International Conference on Research in Computational Molecular Biology*, pages 101–110, 2003.

[63] X Li, X-L Yin, and J Li. Efficient identification of identical-by-descent status in pedigrees with many untyped individuals. *Bioinformatics*, 26(12):i191–i198, 2010.

[64] Louise Lorentsen and Lars Michael Kristensen. Exploiting stabilizers and parallelism in state space generation with the symmetry method. *Application of Concurrency to System Design, International Conference on*, 0:211, 2001.

[65] M S McPeek and T P Speed. Modeling interference in genetic recombination. *Genetics*, 139(2):1031–44, 1995.

[66] M.S. McPeek. Inference on pedigree structure from genome screen data. *Statistica Sinica*, 12(1):311–336, 2002.

[67] M.S. McPeek and L. Sun. Statistical tests for detection of misspecified relationships by use of genome-screen data. *Amer. J. Human Genetics*, 66:1076 – 1094, 2000.

[68] Gilean A. T. McVean, Simon R. Myers, Sarah Hunt, Panos Deloukas, David R. Bentley, and Peter Donnelly. The Fine-Scale Structure of Recombination Rate Variation in the Human Genome. *Science*, 304(5670):581–584, 2004.

[69] C. A. Meacham. A manual method for character compatibility analysis. volume 30, pages pp. 591–600. International Association for Plant Taxonomy (IAPT), 1981.

[70] Ng MY, Levinson DF, and et al. Meta-analysis of 32 genome-wide linkage studies of schizophrenia. *Mol Psychiatry*, 14:774–85, 2009.

[71] S. B. Ng, K. J. Buckingham, C. Lee, A. W. Bigham, H. K. Tabor, K. M. Dent, C. D. Huff, P. T. Shannon, E. W. Jabs, D. A. Nickerson, J. Shendure, and M. J. Bamshad. Exome sequencing identifies the cause of a mendelian disorder. *Nature genetics*, 42(1):30–35, January 2010.

[72] Jurg Ott. *Analysis of Human Genetic Linkage*. Johns Hopkins University Press, 1999.

[73] Laxmi Parida. Ancestral recombinations graph: a reconstructability perspective using random-graphs framework. *Journal of computational biology : a journal of computational molecular cell biology*, 17(10):1227–1252, October 2010.

[74] Itsik Pe'er, Tal Pupko, Ron Shamir, and Roded Sharan. Incomplete directed perfect phylogeny. *SIAM J. Comput.*, 33(3):590–607, 2004.

[75] T. J. Pemberton, C. Wang, J.Z. Li, and N.A. Rosenberg. Inference of unexpected genetic relatedness among individuals in hapmap phase iii. *Am J Hum Genet*, 87(4):457–64, 2010.

[76] A. Piccolboni and D. Gusfield. On the complexity of fundamental computational problems in pedigree analysis. *Journal of Computational Biology*, 10(5):763–773, 2003.

[77] Alkes L. Price, Noah A. Zaitlen, David Reich, and Nick Patterson. New approaches to population stratification in genome-wide association studies. *Nature reviews. Genetics*, 11(7):459–463, June 2010.

[78] S. Purcell, B. Neale, K. Toddbrown, L. Thomas, M. Ferreira, D. Bender, J. Maller, P. Sklar, P. Debakker, and M. Daly. PLINK: A Tool Set for Whole-Genome Association and Population-Based Linkage Analyses. *The American Journal of Human Genetics*, 81(3):559–575, September 2007.

[79] L. Rabiner and B. Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4 – 16, January 1986.

[80] Neil Risch and Kathleen Merikangas. The Future of Genetic Studies of Complex Human Diseases. *Science*, 273(5281):1516–1517, 1996.

[81] S. Sankararaman, G. Obozinski, M.I. Jordan, and E. Halperin. Genomic privacy and limits of individual detection in a pool. *Nature Genetics*, 41(9):965–967, 2009.

[82] C. Semple and M. Steel. *Phylogenetics*. Oxford University Press, 2003.

[83] S. I. Sheikh, T.Y. Berger-wolf, A. A. Khokhar, I. C. Caballero, M. V. Ashley, W. Chaovalitwongse, C. Chou, and B. Dasgupta. Combinatorial reconstruction of half-sibling groups from microsatellite data. *8th International Conference on Computational Systems Bioinformatics (CSB)*, 2009.

[84] Leigh W. Simmons, Rene E C. Firman, Gillian Rhodes, and Marianne Peters. Human sperm competition: testis size, sperm production and rates of extrapair copulations. *Animal Behavior*, 68:297–302, 2004.

[85] E. Sobel and K. Lange. Descent graphs in pedigree analysis: Applications to haplotyping, location scores, and marker-sharing statistics. *American Journal of Human Genetics*, 58(6):1323–1337, 1996.

[86] Yun S. Song, Anand Patil, Erin E. Murphy, and Montgomery Slatkin. Average probability that a cold hit in a dna database search results in an erroneous attribution. *Journal of Forensic Sciences*, 54(1):22–27, 2009.

[87] J. Stankovich, M. Bahlo, J.P. Rubio, C.R. Wilkinson, R. Thomson, A. Banks, M. Ring, S.J. Foote, and T.P. Speed. Identifying nineteenth century genealogical links from genotypes. *Human Genetics*, 117(2–3):188–199, 2005.

[88] M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 1992:91–116, 1992.

[89] Michael R. Stratton, Peter J. Campbell, and P. Andrew Futreal. The cancer genome. *Nature*, 458, 2009.

[90] L. Sun, K. Wilder, and M.S. McPeek. Enhanced pedigree error detection. *Hum. Hered.*, 54(2):99–110, 2002.

[91] B. D. Thatte. Combinatorics of pedigrees, 2006.

[92] B. D. Thatte and M. Steel. Reconstructing pedigrees: A stochastic perspective. *Journal of Theoretical Biology*, 251(3):440 – 449, 2008.

[93] The International HapMap Consortium. The international HapMap project. *Nature*, 426:789–796, 2003.

[94] Alun Thomas. A note on the four-colourability of pedigrees and its consequences for probability calculations. *Statistics and Computing*, 3:51–54, 1993. 10.1007/BF00146954.

[95] E. A. Thompson. *Pedigree Analysis in Human Genetics.* Johns Hopkins University Press, Baltimore, 1985.

[96] T. Thornton and M.S. McPeek. Case-control association testing with related individuals: A more powerful quasi-likelihood score test. *American Journal of Human Genetics*, 81:321–337, 2007.

[97] Cristiano Varin, Nancy Reid, and David Firth. An overview of composite likelihood methods. *Statistica Sinica*, 21:5–42, 2011.

[98] John Wakeley. *Coalescent Theory: An Introduction.* Roberts & Company Publishers, 1 edition, June 2008.

[99] David G. Wang, Jian-Bing Fan, Chia-Jen Siao, Anthony Berno, Peter Young, Ron Sapolsky, Ghassan Ghandour, Nancy Perkins, Ellen Winchester, Jessica Spencer, Leonid Kruglyak, Lincoln Stein, Linda Hsie, Thodoros Topaloglou, Earl Hubbell, Elizabeth Robinson, Michael Mittmann, Macdonald S. Morris, Naiping Shen, Dan Kilburn, John Rioux, Chad Nusbaum, Steve Rozen, Thomas J. Hudson, Robert Lipshutz, Mark Chee, and Eric S. Lander. Large-Scale Identification, Mapping, and Genotyping of Single-Nucleotide Polymorphisms in the Human Genome. *Science*, 280(5366):1077–1082, 1998.

[100] S. Wright. Evolution in Mendelian Populations. *Genetics.*, 16(2):97–159., 1931.

[101] Yufeng Wu. Exact computation of coalescent likelihood under the infinite sites model. In *Proceedings of the 5th International Symposium on Bioinformatics Research and Applications*, ISBRA '09, pages 209–220, Berlin, Heidelberg, 2009. Springer-Verlag.

[102] J. Xiao, L. Liu, L. Xia, and T. Jiang. Efficient Algorithms for Reconstructing Zero-Recombinant Haplotypes on a Pedigree Based on Fast Elimination of Redundant Linear Equations. *SIAM Journal on Computing*, 38:2198, 2009.

[103] K. Zhang and T. Jiang. Some MAX SNP-hard results concerning unordered labeled trees. *Inf. Process. Lett.*, 49(5):249–254, 1994.

Index