1 Haplotype Inference

1.1 Abstract 1-1
1.2 Introduction to Variation, SNPs, Genotypes, and Haplotypes 1-2
The Biological Problem • The Computational Problems • The Need for a Genetic Model • Two Major Approaches
1.3 Clark's Algorithm and Other Rule-Based Methods 1-3
Introduction to Clark's Algorithm • What is the Genetic Model in Clark's Method? • The Maximum Resolution Problem • Improving Clark's Method
1.4 The Pure Parsimony Criterion 1-7
Introduction to Pure Parsimony • A Conceptual Integer Programming Formulation • A More Practical Formulation • Computational Results • Further Work on Pure Parsimony
1.5 Perfect Phylogeny Haplotyping 1-12
Introduction to Perfect Phylogeny Haplotyping • Algorithms and Programs for the PPH Problem • Uniqueness of the Solution: A Phase Transition • Related Models, Results, and Algorithms • Near-Perfect Phylogeny
1.6 Population-Genetic and Statistical Methods 1-19
1.7 Going Forward.. 1-21
1.8 Acknowledgments 1-21

Dan Gusfield
University of California, Davis

Steven Hecht Orzack
Fresh Pond Research Institute

1.1 Abstract

A "haplotype" is a DNA sequence that has been inherited from one parent. Each person possesses two haplotypes for most regions of the genome. The most common type of variation among haplotypes possessed by individuals in a population is the single nucleotide polymorphism (SNP), in which different nucleotides (alleles) are present at a given site (locus). Almost always, there are only two alleles at a SNP site among the individuals in a population. Given the likely complexity of trait determination, it is widely assumed that the genetic basis (if any) of important traits (e.g., diseases) can be best understood by assessing the association between the occurrence of particular haplotypes and particular traits. Hence, one of the current priorities in human genomics is the development of a full *Haplotype Map* of the human genome [1, 46, 47, 17], to be used in large-scale screens of populations [16, 53]. In this endeavor, a key problem is to infer haplotype pairs and/or haplotype frequencies from genotype data, since collecting haplotype data is generally more difficult than collecting genotype data. Here, we review the haplotype inference problem

0-8493-8597-0/01/$0.00+$1.50
© 2001 by CRC Press, LLC

(inferring pairs and inferring frequencies), the major combinatorial and statistical methods proposed to solve these two problems, and the genetic models that underlie these methods.

1.2 Introduction to Variation, SNPs, Genotypes, and Haplotypes

Now that high-throughput genomic technologies are available, the dream of assessing DNA sequence variation at the population level is becoming a reality. The processes of natural selection, mutation, recombination, gene-conversion, genome rearrangements, lateral gene transfer, admixture of populations, and random drift have mixed and remixed alleles at many loci so as to create the large variety of genotypes found in many populations. The challenge is to find those genotypes that have significant and biologically meaningful associations with important traits of interest. A key technological and computational part of this challenge is to infer "haplotype information" from "genotype information". In this section, we explain the basic biological and computational background for this "genotype to haplotype" problem.

In many diploid organisms (such as humans) there are two (not completely identical) "copies" of almost all chromosomes. Sequence data from a single copy is called a haplotype, while a description of the conflated (mixed) data on the two copies is called a genotype. When assessing the genetic contribution to a trait, it may often be much more informative to have haplotype data than to have only genotype data. The underlying data that form a haplotype are either the full DNA sequence in the region, the number of repeats at microsatellite markers, or more commonly the *single nucleotide polymorphisms* (SNPs) in that region. A SNP is a single nucleotide site where more than one (usually two) nucleotides occur with a population frequency above some threshold (often around 5-10%). The SNP-based approach is the dominant one, and high-density SNP maps have been constructed across the human genome with a density of about one SNP per thousand nucleotides [47, 17].

1.2.1 The Biological Problem

In general, it is not easy to examine the two copies of a chromosome separately, and genotype data rather than haplotype data are obtained, although it is the haplotype data that may be of greater use. The data set typically consists of n genotype vectors, each of length m, where each value in the vector is either 0, 1, or 2. The variable n denotes the number of individuals in the sample, and m denotes the number of SNP sites for which one has data. Each site in the genotype vector has a value of 0 (respectively 1) if the associated site on the chromosome has state 0 (respectively 1) on both copies (it is a *homozygous* site); it has a value of 2 otherwise (the chromosome site is *heterozygous*). The goal is to extract haplotype information from the given genotype information.

A variety of methods have been developed and used to do this (e.g., [14, 15, 28, 36, 42, 58, 61, 63, 66, 69]). Some of these methods give very accurate results in some circumstances, particularly when identifying common haplotypes in a population. However, research on haplotype inference continues because no single method is considered fully adequate in all applications, the task of identifying rare haplotypes remains difficult, and the overall accuracy of present methods has not been resolved.

1.2.2 The Computational Problems

The *haplotype inference (HI)* problem can be abstractly posed as follows. Given a set of n genotype vectors, a *solution* to the HI problem is a set of n pairs of binary vectors, one pair for each genotype vector. For any genotype vector g, the associated binary vectors v_1, v_2 must both have value 0 (or 1) at any position where g has value 0 (or 1); but for any position where g has value 2, exactly one of v_1, v_2 must have value 0, while the other has value 1.

A site in g is considered "resolved" if it contains 0 or 1, and "ambiguous" if it contains a 2. If a vector g has zero ambiguous positions, it is called "resolved" or "unambiguous"; otherwise it is called "ambiguous". One can also say that the *conflation* of v_1 and v_2 produces the genotype vector g, which will be ambiguous unless v_1 and v_2 are identical. For an individual with h heterozygous sites there are 2^{h-1} possible haplotype pairs that could underlie its genotype. For example, if the observed genotype g is 0212, then one possible pair of vectors is 0110, 0011, while the other is 0111, 0010. Of course, we want to infer the pair that gave rise to the genotype of each of the n individuals.

A related problem is to estimate the *frequency* of the haplotypes in the sample. We call this the *HF* problem. It is important to note that a solution to the HI problem necessarily solves the HF problem, but the converse is not true.

1.2.3 The Need for a Genetic Model

Non-experimental haplotype inference (the HI and HF problems) would likely be inaccurate without the use of some genetic model of haplotype evolution to guide an algorithm in constructing a solution. The choice of the underlying genetic model can influence the type of algorithm used to solve the associated inference problem.

1.2.4 Two Major Approaches

There are two major approaches to solving the inference problem: combinatorial methods and statistical methods. Combinatorial methods often state an explicit objective function that one tries to optimize in order to obtain a solution to the inference problem. Statistical methods are usually based on an explicit model of haplotype evolution; the inference problem is then cast as a maximum-likelihood or a Bayesian inference problem. Combinatorial approaches are discussed in Sections 1.3 to 1.5.5, and statistical approaches are discussed in Section 1.6.

1.3 Clark's Algorithm and Other Rule-Based Methods

1.3.1 Introduction to Clark's Algorithm

Clark's algorithm to solve the HI problem [14] has been widely used and is still in use today. The algorithm starts by identifying any genotype vectors with zero or one ambiguous sites, since these vectors can be resolved in only one way. These haplotypes are called the *initial resolved* haplotypes; this method requires that some be derivable from the input vectors (sample). One attempts to resolve the remaining ambiguous genotypes by starting with the initial resolved haplotypes. Clark proposed the following rule that infers a new resolved vector *NR* from an ambiguous vector A and an already resolved genotype vector R.

The resolved vector R can either be one of the initial resolved haplotypes, or a haplotype inferred by an earlier application of the following **Inference Rule:**

> Suppose A is an ambiguous genotype vector with h ambiguous sites and R is a resolved vector that is a haplotype in one of the 2^{h-1} potential resolutions of vector A. Then infer that A is the conflation of one copy of resolved vector R and another (uniquely determined) resolved vector *NR*. All of the ambiguous positions in A are set in *NR* to the *opposite* of the entry in R. Once inferred, vector *NR* is added to the set of known resolved vectors, and vector A is removed from the set of ambiguous vectors.

For example, if A is 0212 and R is 0110, then *NR* is 0011.

When the Inference Rule can be used to infer the vector *NR* from the vectors A and R, we say that R can be *applied* to resolve A. It is easy to determine if a resolved vector R can be applied to resolve an ambiguous vector A: R can be applied to A if and only if A and R contain identical unambiguous sites.

Clark's entire algorithm for resolving the set of genotypes is to first identify the initial resolved set, and then repeatedly apply the Inference Rule until either all the genotypes have been resolved, or no further genotypes can be resolved. There are important implementation details for Clark's algorithm that need to be specified, and one can choose different ways to do this. Several alternative variations were studied in [63] and the results of that study will be described in Section 1.3.4.

Note that in the application of the Inference Rule, for any ambiguous vector A there may be several choices for vector R, and any one choice can constrain future choices. Hence, one series of choices might resolve all the ambiguous vectors in one way, while another execution involving different choices might resolve the vectors in a different way, or leave ambiguous vectors that cannot be resolved (orphans). For example, consider two resolved vectors 0000 and 1000, and two ambiguous vectors 2200 and 1122. Vector 2200 can be resolved by applying 0000, creating the new resolved vector 1100, which can then be applied to resolve 1122. In this way, one resolves both of the ambiguous vectors and thereby produces the resolved vector set 0000, 1000, 1100 and 1111. But 2200 can also be resolved by applying 1000, creating 0100. At that point, none of the three resolved vectors, 0000, 1000 or 0100 can be applied to resolve the orphan vector 1122.

Clark's method can produce different solutions depending on how the genotype data are ordered, so the problem of choices is addressed in [14] by reordering the data multiple times and running the algorithm on each ordering. The "best" solution among these executions is reported. Of course, only a tiny fraction of all the possible data orderings can usually be tried. We refer to this as the *local inference method.*

Without additional biological insight, one cannot know which solution (or data ordering) is the most accurate. However, simulations discussed in [14] showed that the inference method tended to produce the wrong vectors only when the execution also leaves orphans. The interpretation is that there is some "global" structure to the set of real haplotype pairs that underlie the observed genotypes, so that if some early choices in the method incorrectly resolve some of the genotypes, then the method will later become stuck, unable to resolve the remaining genotypes. Clark recommended that the execution that resolved the most genotypes should be the best and the one most trusted. The efficacy of the local inference method was shown in [14] and in simulations that we have done, when the data are such that there is a unique execution that maximizes the number of resolved genotypes. However, there are also data sets where most, if not all, of the executions resolve all of the genotypes and do so differently. In that case, some other approach must be used. We discuss that situation in Section 1.3.4.

1.3.2 What is the Genetic Model in Clark's Method?

Here we give a partial justification for the Inference Rule described above.

First, note that Clark's method resolves identical genotypes identically, implying the assumption that the history leading to two identical sequences is identical. The genetic model that justifies this is the "infinite sites" model of population genetics, in which only one mutation at a given site has occurred in the history of the sampled sequences [71]. Second, the Inference Rule seems most applicable when it is assumed that the genotypes in the current population resulted from *random mating* of the parents of the current population. The sampled individuals are also drawn randomly from the population, and the sample is small compared to the size of the whole population, so the initial resolved vectors likely represent *common* haplotypes that appear with high frequency in the population.

These two assumptions are consistent with the way in which the method gives preference to resolutions involving two initially resolved haplotypes. Such haplotypes are always queried first in Clark's method. Similarly, the rule gives preference to resolutions involving one initially resolved haplotype as compared to those involving no initially resolved haplotypes. (However, a logically consistent extension of the rule would require use of two initially resolved haplotypes whenever possible, but this is not what Clark's method does).

We can define the "distance" of an inferred haplotype NR from the initial resolved vectors as the number of inferences used on the shortest path of inferences from some initial resolved vector, to vector NR. The above justification for the use of the Inference Rule becomes weaker as it is used to infer vectors with increasing distance from the initial resolved vectors. However, Clark's Inference Rule is justified in [14] by the empirical observation of consistency discussed above. For additional perspective on Clark's method see [40].

1.3.3 The Maximum Resolution Problem

Given what was observed and proposed in [14], the major open algorithmic question is whether *efficient* rules exist to break choices in the execution of Clark's algorithm, so as to maximize the number of genotypes it resolves. This leads to the **Maximum Resolution (MR) Problem** studied in [35, 36]:

> Given a set of genotypes (some ambiguous and some resolved), what execution maximizes the number of ambiguous vectors that can be resolved by successive application of Clark's Inference Rule?

An algorithm to solve the MR problem must take a more *global* view of the data than does the local inference method, in order to see how each possible application of the Inference Rule influences later choices.

We show in [36] that the MR problem is NP-hard, and in fact, Max-SNP complete. However, the MR problem was reformulated as a problem on directed graphs, with an exponential time (worst case) reduction to a graph-theoretic problem that can be solved via integer linear programming. The general idea is to encode all the possible actions of Clark's algorithm as a directed, acyclic graph. In that graph, each node represents a haplotype that could be generated in some execution of Clark's algorithm, and an edge extends from node u to node v if and only if the haplotype at u can be used to resolve some genotype in the data, resulting in the inference of the haplotype at node v. Accordingly, the MR problem can be formulated as a search problem on this graph, and solved using integer linear programming. Computations [36] showed that this approach is very efficient in practice, and that linear programming alone (without explicit reference to integrality) often suffices to solve the maximum resolution problem. However, an alternative modification of Clark's method proved more successful in obtaining more accurate resolutions. We next discuss this modification.

1.3.4 Improving Clark's Method

Computations done on the MR problem suggest that solving it is not a completely adequate way to find the most accurate solutions. One significant problem is that there are often many solutions to the MR problem, i.e., many ways to resolve all of the genotypes. Moreover, while it is clear that Clark's method should be run many times, and this can generate many different solutions, it is not clear how to use the results obtained. In fact, no published evaluations of Clark's method, except for the evaluation in [63], propose an approach to this issue, and almost all have run Clark's method only once on any given data set. This ignores the stochastic behavior of the algorithm, and these evaluations are uninformative. The critical issue in Clark's method is how to understand and exploit its stochastic behavior.

Clark's method is just one specific instantiation of the "rule-based" approach to the HI problem. In this general approach, one starts by enumerating the unambiguous haplotypes in the sample and then proceeds to use these to resolve ambiguous genotypes. However, different variations of the rule-based approach differ in how the list of reference haplotypes is formed and updated during the process of inferral and how the list of ambiguous genotypes is treated. Some of the variations are discussed in [63]; they can differ in the genetic model of haplotype evolution with which they are consistent.

In [63], we examined the performance of several variations of the rule-based method (including Clark's original method), using a set of 80 genotypes at the human APOE locus, of which 47 were ambiguous; each genotype contained nine SNP sites. The real haplotype pairs were experimentally inferred in order to assess the inferral accuracy of each variation (how many inferred haplotype pairs matched the real haplotype pairs). Most variations produced a large number of different solutions, each of which resolved *all* of the 47 ambiguous genotypes. The variability of accuracy among these solutions was substantial, and a solution chosen at random from among the solutions would likely be one with poor accuracy. Hence, an important issue in using rule-based methods, such as Clark's method, is how to exploit the many different solutions that it can produce.

How to Handle Multiple Solutions: The Consensus Approach

The multiplicity of solutions motivates an effort to understand how they can be used so as to provide a single accurate solution.

We found that the following strategy works to greatly improve the accuracy of any of the variations of the rule-based method. First, for the input genotype data, run the algorithm many times (say, 10,000), each time randomizing the order of the input data. In some variations, we also randomize the decisions that the method makes. The result is a set of solutions that may be quite different from one another. Second, select those runs that produce a solution using the *fewest or close to the fewest* number of *distinct* haplotypes; in our analysis of the APOE data, the number of such runs was typically under 100. For this set of runs, record the haplotype pair that was most commonly used to explain each genotype g. The set of such explaining haplotype pairs is called the "consensus" solution. We observed that the consensus solution had dramatically higher accuracy than the average accuracy of the 10,000 solutions. For example, for the APOE data, out of the 10,000 executions of one of the variations, there were 24 executions that used 20 or 21 distinct haplotypes; no executions used a fewer number of haplotypes. The average accuracy of the 10,000 executions was 29 correct haplotype pairs out of the 47 ambiguous genotypes, and the execution with the highest accuracy in the 10,000 had 39 correct pairs. However, the average accuracy of the 24 selected executions was 36, and the consensus solution of those 24 executions had 39 correct pairs. Hence, this simple rule allowed us to generate a single solution that was as good as the most accurate solution out of all 10,000 solutions. In another variation, the consensus solution had 42 correct pairs, while the average of all the 10,000 solutions had 19 correct pairs. These consensus results compare well with those of other approaches. Multiple executions of the program Phase [69] always produced 42 correct resolutions, whereas the program Haplotyper [61] produced a range of solutions with most getting either 43 or 42 correct, with one solution getting 44 correct, and three solutions getting 37 correct.

We also observed that among the solutions that use the smallest and next-to-smallest number of distinct haplotypes, any haplotype pair that is used with high frequency, say above 85% of the time, was almost always correct. This allows one to home in on those pairs that can be used with high confidence.

1.4 The Pure Parsimony Criterion

1.4.1 Introduction to Pure Parsimony

A different approach to the haplotype inference problem is called the *Pure-Parsimony* approach. To our knowledge, this approach was first suggested by Earl Hubbell, who also proved that the problem of finding such solutions is NP-hard [49]. The Pure-Parsimony problem is:

> Find a solution to the haplotype inference problem that minimizes the total number of distinct haplotypes used.

For example, consider the set of genotypes: 02120, 22110, and 20120. There are solutions for this example that use six distinct haplotypes, but the solution (00100, 01110), (01110, 10110), (00100, 10110) uses only three distinct haplotypes.

Use of such a parsimony criterion is consistent with the fact that the number of distinct haplotypes observed in most natural populations is vastly smaller than the number of possible haplotypes; this is expected given the plausible assumptions that the mutation rate at each site is small and recombinations rates are low. Further, we observed in Section 1.3.4 that the most accurate rule-based solutions were those that inferred a small number of distinct haplotypes.

We note that some authors have described Clark's method [14] as relying on a parsimony criterion for the number of haplotypes used [2, 61], although there is no such criterion in the method and in fact, it rarely produces a most parsimonious solution in our experience (see Section 1.3.4). The inferral program Phase [69] has also been described as relying on a parsimony criterion [22]. However, the complex details of its computation makes it hard to quantify the influence of a parsimony criterion. This makes it difficult to use any of these methods to evaluate the effectiveness of the *parsimony criterion* as an objective function in solving the HI problem.

In [38] we described how to use integer linear programming to compute an HI solution that minimizes the number of distinct haplotypes, i.e., solving the Pure-Parsimony problem. However, the theoretical worst-case running time of this method increases exponentially with the number of genotypes, so empirical studies were undertaken to see if this approach is practical for data sets of current interest in population-scale genomics. The basic approach, combined with additional ideas presented in [38], is practical on moderately sized datasets, and this allows a comparison of the accuracy of solutions based on the Pure-Parsimony criterion and the accuracy of solutions obtained from inferral methods not so based. This is detailed in the next section.

1.4.2 A Conceptual Integer Programming Formulation

We begin by describing a *conceptual* integer-linear-programming solution to the Pure-Parsimony problem. The solution would generally be impractical to use without additional improvements. After describing this solution, we introduce two simple observations that make it practical for data sets of current biological interest.

Let g_i denote the ith genotype input vector, and suppose it has h_i ambiguous sites. There are 2^{h_i-1} pairs of haplotypes that could have generated g_i. We enumerate each one of these pairs, and create one integer programming variable $y_{i,j}$ for each of the 2^{h_i-1} pairs. As we create these y variables, we take note of the haplotypes in the enumerated pairs. Whenever a haplotype is enumerated that has not been seen before, we generate a new integer programming variable x_k for that haplotype. There will only be one x variable generated for any given haplotype, regardless of how often it is seen in a genotype.

What are the linear programming constraints? Consider the following example. For genotype $g_i = 02120$ we enumerate the two haplotype pairs (00100, 01110) and (01100, 00110), and generate the two variables $y_{i,1}$ and $y_{i,2}$ for these pairs. Assuming that these four haplotypes have not been seen before, we generate the four variables x_1, x_2, x_3, x_4 for them. We create the constraint

$$y_{i,1} + y_{i,2} = 1$$

All of the x and y variables can be set only to 0 or 1. Therefore, this inequality says that in a solution, we must select *exactly* one of the enumerated haplotype pairs as the resolution of genotype g_i. Which y variable in this constraint is set to 1 indicates which haplotype pair will be used in the explanation of genotype g_i.

Next, we create two constraints for *each* variable $y_{i,j}$. In our example, these are:

$$y_{i,1} - x_1 \leq 0$$

$$y_{i,1} - x_2 \leq 0$$

$$y_{i,2} - x_3 \leq 0$$

$$y_{i,2} - x_4 \leq 0$$

The first constraint says that if we set $y_{i,1}$ to 1, then we must also set x_1 to 1. This means that if we select the haplotype pair associated with variable $y_{i,1}$ to explain g_i, then we must use the haplotype associated with variable x_1, because that haplotype is one of the pair of haplotypes associated with variable $y_{i,1}$. The second constraint says the same thing for the haplotype associated with variable x_2.

These are the types of constraints that are included in the integer programming formulation for *each* input genotype. If a genotype has h ambiguous sites, then there will be exactly $2^h + 1$ constraints generated for it.

For the objective function, let X denote the set of all the x variables that are generated by the entire set of genotypes. Recall that there is one x variable for each distinct haplotype, no matter how many times it occurs in the enumerated pairs. Then the objective function is:

$$\text{Minimize} \sum_{x \in X} x$$

This function forces the x variables to be set so as to select the *minimum* possible number of distinct haplotypes. Taken together, the objective function and the constraints, along with the restriction that the variables can only be set to 0 or 1, specify an integer-linear-programming formulation whose solution minimizes the number of distinct haplotypes used. Thus, this formulation solves the "Pure-Parsimony" haplotype problem. This formulation is called the "TIP formulation" in [38].

1.4.3 A More Practical Formulation

For many current data sets (50 or more individuals and 30 or more sites) the large number of constraints generated in the TIP formulation make it impractical to solve the resulting integer program. For that reason, additional ideas are required to make it practical.

The first idea is the following: if the haplotype pair for variable $y_{i,j}$ consists of two haplotypes that are both part of *no* other haplotype pair, then there is no need to include variable $y_{i,j}$ or the two x variables for the two haplotypes in the pair associated with $y_{i,j}$ in the integer program. In this way, we create a "reduced" formulation by removing such y and x variables. This formulation is called the "RTIP" formulation in [38].

The RTIP formulation correctly solves the Pure-Parsimony problem because if there is a genotype vector g such that all associated y variables are removed from the TIP formulation, then there is an optimal solution to the TIP formulation where we arbitrarily choose a permitted haplotype pair for g. Otherwise, there is an optimal solution to the TIP formulation that does not set any of the removed x or y variables to 1. Hence, there is no loss in removing them, and the RTIP formulation will find the same solution that the TIP formulation finds.

This reduced formulation is particularly effective because DNA sequences in populations have generally undergone some amount of recombination, a process that creates two chimeric sequences from two input sequences. Depending on the realized level of recombination in the evolution of the sequences, the reduced formulation can be much smaller (fewer variables and inequalities) than the original original formulation. The reason is that as the level of recombination increases, the number of distinct haplotypes in the sample typically increases and the frequency distribution of the haplotypes becomes more uniform. Therefore, more of the haplotypes appear only in one of the sampled genotypes. These "private" haplotypes are removed in the RTIP formulation. Smaller formulations generally allow integer programming solution codes to run faster.

The reduced formulation preserves the optimal solution to the original formulation. However, if the reduced formulation is created by first creating the original formulation and then removing variables and constraints, the work involved could still make this approach impractical. The following is a more efficient way to create the reduced formulation: let g_i be a genotype vector, and let H_i be the set of haplotypes that are associated with g_i in the original integer programming formulation. Then for any pair of genotypes g_i, g_j, it is easy to identify the haplotypes in $H_i \cap H_j$, and to generate them in time proportional to $m|H_i \cap H_j|$, where m is the length of the genotype vector. Simply scan g_i and g_j left to right; if a site occurs with a value of 1 in one haplotype and 0 in the other, then $H_i \cap H_j = \emptyset$; if a site occurs with a 2 in one vector and a 0 or 1 in the other, then set that 2 to be equal to the other value. Then if there are k remaining sites, where both g_i and g_j contain 2's, there are exactly 2^k distinct haplotypes in $H_i \cap H_j$, and we generate them by setting those k sites to 0 or 1 in all possible ways. The time for this enumeration is proportional to $m|H_i \cap H_j|$. Moreover, each generated haplotype in $H_i \cap H_j$ specifies a haplotype pair that will be included in the reduced formulation, for both g_i and g_j.

Any x variable that is included in the reduced formulation must occur in an intersecting set for some pair of genotypes, and every pair of haplotypes that should be associated with a y variable must also be found while examining some pair of genotypes. Hence, the reduced formulation can be produced very quickly if it is small.

1.4.4 Computational Results

Computations reported in [38] show that a Pure Parsimony solution for problem instances of current interest *can* be efficiently found in most cases. The practicality and accuracy of the reduced formulation depend on the level of recombination in the data (the more recombination, the more practical but less accurate is the method). We show in [38] that the Pure-Parsimony approach is practical for genotype data of up to 50 individuals and 30 sites. Up to moderate levels of recombination, 80 to 95 percent of the inferred haplotype pairs are correct, and the solutions are generally found in several seconds to minutes, except for the no-recombination case with 30 sites, where some solutions require a few hours.

When the recombination rate is low, Pure-Parsimony solutions were generally as accurate as those obtained with the program Phase [69]. However, they become somewhat inferior to Phase solutions when the recombination rate becomes large. Nonetheless, these Pure Parsimony results are a validation of the genetic model implicit in the Pure-Parsimony objective function, for a randomly picked solution would correctly resolve only a minuscule fraction of the genotypes. It is conceivable that a program that adds heuristics to the Pure Parsimony criterion would produce results that are competitive with programs such as Phase.

1.4.5 Further Work on Pure Parsimony

The Pure-Parsimony criterion has also been examined in [74], where a branch-and-bound method, instead of integer programming, was used to solve the problem. More theoretical results on pure parsimony appear in [40, 48, 54, 55]. No computational results on pure parsimony are reported in these papers. The first two papers also presented an integer linear programming formulation of the problem whose size grows polynomially with the size of the input. This is in contrast with the approach in [38] where (in the worst case) the size of the integer program can grow exponentially with the size of the input (although in practice, the growth is more modest).

A Polynomial-size Integer Linear Programming (ILP) Formulation for Pure Parsimony

A different polynomial-size integer linear programming formulation was developed [10] along with additional inequalities (cuts) that decrease the running time needed to solve the integer program. This formulation was also presented in [41] without the additional inequalities, and without computational results.

In this ILP formulation, for each genotype vector i, we create two binary variables (which can take on values 0 or 1 only), $y(2i-1,j)$ and $y(2i,j)$, for each site j in genotype vector i. If site j in genotype i is homozygous with state 0, then we create the constraint:

$$y(2i-1,j) + y(2i,j) = 0$$

If site j in genotype i is homozygous with state 1, then we create:

$$y(2i-1,j) + y(2i,j) = 2$$

If site j in genotype i is heterozygous, then we create:

$$y(2i-1,j) + y(2i,j) = 1$$

For any genotype vector i, the states of the variables $y(2i-1,j)$ and $y(2i,j)$, for all m sites, should define two haplotypes that explain genotype i. The above constraints ensure that any solution to the ILP creates two haplotypes that explain each genotype.

We want to minimize the number of distinct haplotypes used, and the key issue is how to set up constraints to do this. As a first step, let k and $k' < k$ be any two indices between 1 and $2n$, i.e., indices for the $2n$ haplotypes produced by a solution. We will use "haplotype k" to denote the haplotype indexed by k in the solution. For each (k', k) pair, we create the variable $d(k', k)$, which we want to be set to 1 if (but not only if) haplotype k' is different from haplotype k. This is accomplished by creating the following two constraints for each site j:

$$d(k',k) \geq y(k,j) - y(k',j)$$
$$d(k',k) \geq y(k',j) - y(k,j)$$

The variable $d(k', k)$ will be set to 1 if haplotypes k and k' are different, since they will be different if and only if they differ in at least one site j.

We next introduce the variable $x(k)$, for each k from 1 to $2n$, which we want to be set to 1 in a solution, if (but not only if) haplotype k is distinct from all of haplotypes $k' < k$. This is achieved with the following constraint:

$$[\sum_{k'=1}^{i-1} d(k',k)] - i + 2 \leq x(k)$$

To understand this constraint, note that if haplotype k is different from every haplotype $k' < k$, then $\sum_{k'=1}^{i-1} d(k', k) = i - 1$, and so

$$[\sum_{k'=1}^{i-1} d(k',k)] - i + 2$$

will equal one.

With the above constraints, a solution to this integer program specifies a pair of haplotypes that explain the genotypes, where $\sum_{k=1}^{2n} x(k)$ is greater than or equal to the number of distinct haplotypes in the solution. Therefore, by using the objective function

$$Minimize \sum_{k=1}^{2n} x(k),$$

any solution to this integer program will be a solution to the Pure Parsimony problem.

The reader can verify that the number of variables and constraints grows only polynomially with n and m, rather than exponentially (in worst case) as in the TIP and RTIP formulations.

No computation were shown in [41], but extensive computations shown in [10] compared the polynomial-size formulation with the earlier formulation in [38]. Perhaps surprisingly, the exponential-size formulation did not always run slower than the polynomial-size formulation, and there were many cases where the former formulation ran in seconds while the latter formulation took hours (although there were cases where the opposite was observed). Perhaps the reason is that smaller formulation has to computationally discover necessary features of the optimal solution (such as the candidate haplotype pairs) that are explicitly specified in the larger formulation.

Recent Contributions

More recently, a hybrid formulation that combines ideas from [38] and [10] was developed and tested in [11]. The result is an integer programming formulation that again only uses polynomial space (similar to the formulation in [10]), but whose running time in practice is closer to the running time observed with the RTIP formulation, although it is still generally slower than that formulation. The hybrid formulation allows practical computation of problem instances whose RTIP formulation is too large to fit into memory, and whose running time with the formulation from [10] is excessive.

In a somewhat different direction, an approximation algorithm was developed and tested in [48] using Semidefinite Programming Relaxation of an Integer Quadratic Programming formulation of the Pure Parsimony problem. This method was shown to compare well in both speed and accuracy with several other haplotyping methods when applied to simulated and real data sets. Other recent work on Pure-Parsimony includes a heuristic algorithm that builds a solution in a somewhat greedy manner [57].

1.5 Perfect Phylogeny Haplotyping

1.5.1 Introduction to Perfect Phylogeny Haplotyping

As noted earlier, the haplotype inference problem would be impossible to solve without some implicit or explicit genetic assumptions about how DNA sequences evolve. An important set of such assumptions are embodied in the population-genetic concept of a *coalescent* [50, 71]. A coalescent is a stochastic process that provides an evolutionary history of a set of sampled haplotypes. This history of the haplotypes is represented as a directed, acyclic graph, where the lengths of the edges represent the passage of time (in number of generations). In our problems, we ignore time, so we are only concerned with the fact that the history is represented by a directed, acyclic graph. The key observation [50] is that “In the absence of recombination, each sequence has a single ancestor in the previous generation.” Hence, if we trace back the history of a single haplotype H from a given individual I, we see that haplotype H is a copy of one of the haplotypes in one of the parents of individual I. It doesn’t matter that I had two parents, or that each parent had two haplotypes. The backwards history of a single haplotype in a single individual is a simple path, if there is no recombination. That means the histories of two sampled haplotypes (looking backwards in time) from two individuals merge at the most recent common ancestor of those two individuals.

There is one additional element of the basic coalescent model: the *infinite-sites* assumption (see above). This assumption is justified when the probability of mutation at any given site is small, so that the probability of two or more mutations at a given site can be taken as zero. Hence, the coalescent model of haplotype evolution says that without recombination, the true evolutionary history of $2n$ haplotypes, one from each of $2n$ individuals, can be displayed as a tree with $2n$ leaves, and where each of the m sites labels exactly one edge of the tree.

More formally, if M is a set of binary sequences, and V is a binary sequence that will label the root, the tree displaying the evolution of the haplotypes is called a *perfect phylogeny for M and V* [33, 34]. It is a rooted tree T with exactly $2n$ leaves that obeys the following properties:

1. The root of T is labeled with an m-length binary vector V, which represents the “ancestral sequence”, i.e., the ancestral state of each of the m sites.

2. Each of the $2n$ rows labels exactly one leaf of T, and each leaf is labeled by one row.
3. Each of the m columns labels *exactly one* edge of T.
4. Every interior edge (one not touching a leaf) of T is labeled by *at least* one column.
5. For any row i, the value $M(i, j)$ is unequal to $V(j)$ if and only if j labels an edge on the unique path from the root to the leaf labeled i. Hence, that path, relative to V, is a compact representation of row i.

Often we assume that V is the all-zero vector, but the above definition is more general.

An illustration of a perfect phylogeny and of its use in "association mapping" are presented in [3].

Part of the motivation for the perfect phylogeny model (i.e., coalescent without recombination) comes from recent observations [18, 73] of little or no evidence for recombination in long segments of the Human genome, and the general belief that most SNPs are the result of a mutation that has occurred only once in human history [47].

Formally, the **Perfect Phylogeny Haplotype (PPH) Problem** is:

> Given a set of genotypes, M, find a set of explaining haplotypes, M', which defines a perfect phylogeny.

In the perfect phylogeny model, each genotype vector (from a single individual in a sample of n individuals) was obtained from the mating of two of $2n$ haplotype vectors in an (unknown) coalescent (or perfect phylogeny). In other words, the coalescent with $2n$ leaves is the history of haplotypes in the $2n$ *parents* of the n individuals under study. Those $2n$ haplotypes are partitioned into pairs, each of which gives rise to one of the n genotypes.

So, given a set S of n genotype vectors, we want to find a perfect phylogeny T, and a pairing of the $2n$ leaves of T that explains S. In addition to efficiently finding one solution to the PPH problem, we would like to determine if that is the *unique* solution, and if not, we want to represent the set of *all* solutions, so that each one can be generated efficiently.

1.5.2 Algorithms and Programs for the PPH Problem

The PPH problem was introduced and first solved in [37], where it was explained that after one PPH solution is obtained, one can build an implicit representation of the set of all PPH solutions in $O(m)$ time. The algorithm given in [37] is based on reducing the PPH problem to a well-studied problem in graph theory, called the *graph-realization* problem. The theoretical running time of this initial approach is $O(nm\alpha(nm))$, where α is the inverse Ackerman function, usually taken to be a constant in practice. Hence, the worst-case time for the method is nearly linear in the size of the input, nm. The time for the reduction itself is $O(nm)$, and the graph-realization problem can be solved by several published methods. In [37] we used a graph-realization algorithm (the Bixby-Wagner algorithm) [8] in order to establish the near-linear time bound for the PPH problem. The Bixby-Wagner algorithm is based on a general algorithm due to Löfgren [59], and runs in $O(nm\alpha(nm))$ time. However, the Bixby-Wagner algorithm is difficult to understand and to implement. Accordingly, we implemented a reduction-based approach using a different solution to the graph-realization problem [30]. The resulting program (called GPPH) [13] has a worst-case running time of $O(nm^2)$. Recently, the original reduction-based approach was implemented [56] using a Java implementation of the Bixby-Wagner method [62, 51].

A second program to solve the PPH problem (called DPPH) is based on deeper insights into the combinatorial structure of the PPH problem, rather than on a reduction to the graph-realization problem. The algorithm underlying DPPH was developed in [5]. The running time for the algorithm and program is also $O(nm^2)$, and the algorithm produces a graph that represents all solutions in a simple way. Insights similar to those in [5] were presented in [77], but were not developing into an explicit algorithm for solving the PPH problem.

A third algorithm to solve the PPH problem was developed in [25], and it has been implemented in a program we call BPPH [12]. That algorithm and program also have worst-case running time of $O(nm^2)$, and they can be used to find and represent all solutions.

A Linear-Time Solution

Recently, we developed an algorithm for the PPH problem that runs in $O(nm)$ time, i.e., in *linear* time [21]. The program based on this algorithm is called LPPH. An alternative linear-time algorithm was published in [67]. The results of empirical testing of the first three programs mentioned above can be found in [12]. Some comparisons of LPPH to DPPH (the fastest of the first three) are also shown in [21]. LPPH is significantly faster than DPPH when the number of sites is large. For example, in tests with $n = 1000$ individuals and $m = 2000$ sites, DPPH ran for an average of 467 seconds, while LPPH ran for an average of 1.89 seconds. All four of the PPH programs can be obtained at wwwcsif.cs.ucdavis.edu/~gusfield/.

The conceptual and practical value of a linear-time solution can be significant. Although most current inference problems involve under one hundred sites, where the differences in running time between the programs are not of great practical significance, there are regions of the human genome up to several hundred kilobases long where the SNP states are highly correlated. Such high correlation is called "linkage disequilibrium" (LD), and high LD suggests that little or no recombination has occurred in those regions. Further, there is very little known about haplotype structure in populations of most organisms, so it is too early to know the full range of *direct* application of this algorithm to PPH problems involving long sequences (see [12] for a more complete discussion).

Faster algorithms are of practical value when the PPH problem is repeatedly solved in the inner-loop of an algorithm. This occurs in the inference of haplotype pairs affected by recombination [70], and when searching for recombination hotspots and low-recombination blocks [77]. In both of these cases one finds for every SNP site the longest interval starting at that site for which there is a PPH solution. When applied on a genomic scale (as is anticipated), even a ten-fold increase in speed is important. Moreover, in some applications, one may need to examine *subsets* of the given SNP sites for which there is a PPH solution. This is partly due to small departures from the perfect phylogeny model. It is also motivated by observations of subsets of SNP sites with high pairwise LD, where the sites in the subset are not contiguous in the SNP haplotype, but are are interlaced with other SNP sites which are not in high LD with sites in the subset. Such a subset of SNP sites is called a *dispersed haplotype block*. The lengths of these dispersed haplotype-blocks are not known. When solving the PPH problem repeatedly on a large number of subsets of sites, increased efficiency in the inner loop will be important, even if each subset is relatively small.

The High Level Idea Behind the Linear-Time Solution

In obtaining the linear-time solution [21], we used the general method of Löfgren, but we exploited properties of the PPH problem to obtain a specialized version that is simpler to implement than the Bixby-Wagner graph-realization method. Although there is no explicit mention of the graph-realization problem in [21], in order to develop the intuition behind the method, it is useful to review a bit of the Whitney-Löfgren theory of graph-realization, specialized to the PPH problem.

Let M be an instance of the PPH problem, and let T be a perfect phylogeny that solves the PPH problem for M. Each leaf of T is labeled by one row of M, and each row of M labels *two* distinct leaves of T. We define a *three-partition* of the *edges* of T to be a partition of the edges of T into three connected, directed subtrees T_1, T_2, T_3 of T, such that T_1 is rooted at the root of T, and T_2 and T_3 are rooted at distinct nodes, u, v (respectively) in T_1. Note T_1 might consist only of the root node of T; also note that the *nodes* of T are not partitioned between the three subtrees, and that either, but not both, of u or v might be the root of T. A *three-partition* is *legal* if the set of labels of the leaves in T_2 is identical to the set of labels of the leaves in T_3. Given a legal three-partition, we define a *legal flip* in T of T_2 and T_3 as the following operation: disconnect trees T_2 and T_3 from T, and merge node u in T_2 with node v in T_1, and merge node v in T_3 with node u in T_1.

The application of Whitney's theorem [76] to the PPH problem implies that *every* PPH solution for M can be obtained by a series of legal flips, starting from any PPH solution T, and *every* tree T' created in the series is also a PPH solution for M. Moreover, the number of needed flips is bounded by the number of edges of T. This theorem is the basis for Löfgren's graph-realization algorithm, and the version of the theorem above specializes to a version of Löfgren's algorithm that solves the PPH problem. We will describe this approach for the case when M contains only entries that are 0 or 2. The effect of having no 1 entries is that for every row i, and every PPH solution T, the path in T between the two leaves labeled i must go through the root of T.

We now describe at a high level the approach to solving the PPH problem based on Löfgren's general method for solving the graph-realization problem. Let $T(k)$ be a solution to the PPH problem restricted to the first k rows of M. Let $OLD(k+1)$ be the set of sites that have value 2 in row $k+1$ and have value 2 in some row 1 through k. Each site in $OLD(k+1)$ is an "old" site, and already labels an edge in $T(k)$. Let $N(k+1)$ be the remaining set of sites in row $k+1$, i.e., the "new" sites. Let $M'(k+1)$ be the matrix made up of the first k rows of M, together with a new row created from row $k+1$ by setting to 0 all the entries in $N(k+1)$. Whitney's theorem implies that if there is a PPH solution for $M'(k+1)$, then a PPH solution for $M'(k+1)$ can be obtained by a series of legal flips (each relative to a three-partition) starting from $T(k)$. Löfgren's algorithm finds such a solution $T'(k+1)$ for $M'(k+1)$ by finding an appropriate series of flips. Moreover, there is a series of flips that can find a particular solution $T'(k+1)$, so that the sites in $N(k+1)$ can be added in a single path, at the end of one of the two paths in $T'(k+1)$ that contain the sites of $OLD(k+1)$.

Additional ideas and appropriate data structures are needed to make this approach efficient. A key idea is that when finding a solution $T'(k+1)$, we never allow a flip that is forced to be done later in the opposite direction, and so all the edges incident with the nodes u and v (defined in the three-partition) can be "fixed", thus specifying more of the PPH solution for M (if there is a solution). At each point in the execution of the algorithm, a data structure called a "shadow tree" implicitly represents all possible solutions to the problem seen so far. As the algorithm proceeds, more of the solution becomes fixed, and the shadow tree at the end of the algorithm represents all solutions to the PPH problem.

1.5.3 Uniqueness of the Solution: A Phase Transition

For any given set of genotypes, it is possible that there will be more than one PPH solution. How many individuals should be in the sample so that the solution is very likely to be unique? To answer this question, we did computations that determine the frequency of a unique PPH solution for various numbers of sites and of genotypes [12]. Intuitively, as the ratio of genotypes to sites increases, one expects that the frequency of unique solutions should increase. This was observed, as was a striking *phase transition* in the frequency of unique solutions as the number of individuals grows. In particular, the frequency of unique solutions is close to zero for a small number of individuals, and then jumps to over 90% with the addition of just a few more individuals. In our computations, the phase transition occurs when the number of individuals is around twenty-five. The phase transition was also found in computations done by T. Barzuza and I. Pe'er [7], although they observed the transition with somewhat fewer individuals than in our computations. These results have positive practical implications, since they indicate that a surprisingly small number of individuals is needed before a unique solution is likely.

1.5.4 Related Models, Results, and Algorithms

The PPH problem has become well-known (see the surveys [9, 39, 40, 41]), and there is now a growing literature on extensions, modifications, and specializations of the original PPH problem [4, 6, 19, 20, 26, 25, 42, 45, 52] and on the PPH problem when the data or solutions are assumed to have some special form [31, 32, 43]. Some of these papers give methods that run in linear time, but only work for special cases of the PPH problem [31, 32], or are only correct with high probability [19, 20]. Some of the papers discuss the problems associated with incomplete or incorrect data, some develop complexity results that limit the extent that one can expect to obtain polynomial-time methods, and some consider different biological contexts that change some of the details of the problem. We will now discuss some of these results.

Papers by [31, 52] showed that the the PPH problem is NP-complete when the data are incomplete. It was established in [4] that the problem of finding a PPH solution that *minimizes* the number of distinct haplotypes it uses is NP-hard. It was also established there that the $O(nm^2)$-time solutions to the PPH problem in [5, 25] are unlikely to be implementable in $O(nm)$ time, even though the same paper shows that if either method could be implemented in $O(nm + m^2)$ time, then the algorithm could be implemented in $O(nm)$ time. The PPH solution in [5] runs in $O(nm)$ time, except for an initial computation that runs in $O(nm^2)$ time but only produces output of size $O(m^2)$. So it seemed attractive to see if that initial computation could be implemented to run in $O(m^2)$ time. The method in [25] contains the same initial computation, and although no explicit algorithm is presented in [77], the ideas there are based on this initial computation. However, we showed in [4] that the initial computational task is equivalent to boolean matrix multiplication. That implies that if the computation could be implemented to run in $O(nm)$ time, then two n by n boolean matrices could be multiplied in $O(n^2)$ time, which is significantly faster than is currently possible.

He and Zelikovsky [45] used linear algebra to find redundancies that can be removed to reduce the number of sites in an instance of the HI problem. This approach is not guaranteed to preserve the set of solutions, but the typical loss of accuracy can vary depending on which specific haplotyping method is used. When tested along with the program DPPH (solving the PPH problem), this approach resulting in little loss of accuracy and a large increase in speed.

In contrast to papers that focus primarily on algorithmic issues related to the PPH problem, several papers discuss variants of the original PPH problem that arise in different biological contexts. The papers [31, 32] considered the PPH problem where the input is assumed to have a row where all the entries have value two. That is, there must be a pair of haplotypes in the solution in which every site is heterozygous. Such a pair is called a "yin-yang" haplotype; they are found in many populations [79]. Hence, in any solution T to a PPH problem of this type, there must be two paths from the root of T that contain all of the sites. Note that there may be rows that are not all-2, but since a solution T must have two directed paths from the root that contain all of the sites, any other haplotype in the solution must be defined by a path in T that forms some initial portion of one of those two paths. Thus, this variant of the PPH problem is called the "Perfect Phylogeny Path Haplotyping" (PPPH) problem. The method in [31] is simple and runs in linear time. The PPPH problem may be related to the classical "consecutive ones" problem. Part of the intuition for this is that the graph-realization problem, which can be viewed as the basis for the PPH problem, is a generalization of the consecutive-ones problem, where the "ones" have to form consecutive intervals in a tree instead of on the line. But the PPPH problem is the PPH problem when restricted to a single path, which can be embedded on a line.

The XOR PPH problem

Suppose that the genotype vector for an individual indicates whether a site is heterozygous or homozygous, but does not indicate the specific state of a homozygous site. Genotype vectors of this type may be cheaper and easier to obtain than those that indicate the specific state at every homozygous site. Such a genotype vector is the XOR (exclusive OR) of the two (0-1)-haplotype vectors [6].

Given a set of n such XOR genotypes, the XOR PPH problem is to find a perfect phylogeny T with $2n$ leaves, and a pairing of the leaf sequences of T, so that the n input genotype vectors result from taking the XOR of each of the n paired leaf sequences. The main result in [6] is that this problem can also be reduced to an instance of the graph-realization problem, as in the original PPH problem, and hence can be solved in $O(\alpha(nm)nm)$ time in theory. As in the PPH problem, initial implementations were based on using a slower and simpler solution to the graph-realization problem, resulting in an $O(nm^2)$-time algorithm for the XOR PPH problem. Just as in the original PPH problem, it is important to assess how many individuals are needed in the sample in order to find a PPH solution (if there is one) that is likely to be unique. Computations were done in [6] to compare the number of needed individuals in the two PPH formulations, and the result is that a high probability of uniqueness is obtained for XOR genotype input using only a few more individuals than with the full PPH genotype input.

There is another interesting and potentially important result in [6] concerning the PPH model and the so-called Tag SNPs in "haplotype blocks". Several recent studies have found long regions in human DNA, called haplotype blocks, where high LD is observed (see [73] for a review). There are other definitions of haplotype blocks that are not explicitly based on LD, such as defining a block to be a region of sufficient length for which only a small number of haplotypes are found among most individuals. (Different instantiations of the words "sufficient", "most", and "few" lead to different precise block definitions and to different methods to recognize blocks or to partition a sequence into blocks.) No matter what the causal basis is for haplotype blocks, they can be exploited to make large-scale genotyping more practical. The high association between the states of SNP sites inside a single haplotype block makes it possible to identify a small number of "Tag-SNP" sites in a set of haplotypes, whose states act as a label for all (or most) of the haplotypes in the sample. In other words, the (0-1) states of the Tag-SNPs for an individual allow one to determine the states of the other SNP sites for that individual. Given the haplotypes in a sample, a smallest set of Tag-SNPs can be found by solving an instance of a "minimal test-set problem", which is easily framed as a set-cover problem. The minimal test-set problem is NP-hard, but is solvable in practice for current data sets (up to a few hundred individuals and one hundred sites). The advantage of knowing a small set of Tag-SNPs is clear: if the haplotypes in yet unstudied individuals are like those in the sampled individuals, one would need to look only at the Tag-SNPs to infer the haplotype pairs of the unstudied individuals.

The definition of a Tag-SNP has been for haplotypes, but it is *genotypes* that will be determined in large screens. Can one find a subset of sites in the genotypes of the sample, so that for any individual in the sample, the values at those genotypes determine the two underlying haplotypes of the individual? One can define and identify "Tag genotype SNPs" as a set of sites that determine the genotype values at the other sites in the sample. But is a set S of Tag genotype SNPs also a set of Tag-SNPs for that underlying haplotype solution in the sample? If so, by knowing the genotype values at S, one would know all the genotype values for the individual, and *also* know the underlying haplotype pair for that individual. It is shown in [6] that a set of Tag genotype SNPs is *not* always a set of Tag haplotype SNPs, but when the data have a PPH solution (or a XOR PPH solution if only XOR genotypes are known) a set of Tag genotype SNPs is also a set of Tag haplotype SNPs. In this case, genotype data in large screens are as useful as haplotype data. This is potentially a very important result.

1.5.5 Near-Perfect Phylogeny

One modification of the PPH problem, called the "imperfect" or "near-perfect" or "almost-perfect" phylogeny haplotyping problem deserves particular attention. This approach was developed in three papers by E. Eskin, E. Halperin, and R.M. Karp [24, 25, 42], and is implemented in a program called HAP [42]. HAP was recently used to infer haplotype pairs and to predict haplotype-blocks, in the largest-yet published study of the patterns of SNP variation in human populations [47].

The main motivation for the near-perfect phylogeny model is the observation that in certain well-studied data sets (e.g., [18]), the common haplotypes (the ones most frequently seen in the sample) fit the perfect phylogeny model, but the full set of haplotypes in the sample do not. Halperin and Eskin [42] stated that "infrequent haplotypes cause the majority of the conflicts with the perfect phylogeny model". They derived this conclusion from studying the data in [18], where very few conflicts remain after the removal of haplotypes that occur in fewer than 5% of the sample, and no conflicts remain after the removal of haplotypes that occur in fewer than 10% of the sample. Thus, the haplotypes fit the perfect-phylogeny model after modifications are made to the data, and are said to fit a "near-perfect-" or "almost-perfect-" phylogeny.

HAP uses this observation to perform haplotype inference in non-overlapping fixed-length intervals of contiguous SNP sites. In each such interval, the program finds a subset of the genotypes (possibly all of them) for which there is a solution to the HI problem that fits a perfect phylogeny. These haplotypes are expected to be the common haplotypes in the population. It then uses these haplotypes to infer haplotype pairs for the remaining genotypes, which may include genotypes initially removed due to missing data. Missing values in a genotype are inferred from the haplotypes in the PPH solution by a maximum-likelihood approach.

We now discuss how HAP finds haplotype pairs for the common haplotypes in a fixed interval. The program derives from an algorithm [24] that solves the original PPH problem. The specific ways that HAP modifies that algorithm and the ways that it modifies the data as the algorithm proceeds have not been completely explained in [42] (HAP contains well over 10,000 lines of code). But the general ideas have been articulated as follows [42]. The PPH algorithm in [24] builds a PPH solution from the root of the tree downward, examining each row of the input in turn. When examining a row, it may learn that in all PPH solutions, a specific pair of sites must be together on a directed path from the root, or it may learn the opposite, that they cannot be together on any directed path from the root. Alternatively, the examination of a row may indicate that there is no PPH solution possible. HAP follows the general outline of this algorithm, but instead of acting under the influence of a single row, it looks at additional rows to see how many rows support the same conclusion (for example, that the edge containing one of the sites should be made an ancestor of the edge containing the other). If only a small number of rows support that conclusion, and the other rows support an alternative conclusion, then the minority action is not taken and the (few) rows supporting the action can be removed (this detail is not explicitly stated in [42]). In this way, a perfect phylogeny is created for a subset of the rows. One of the features of the algorithm in [24] (and other algorithms for the PPH problem) is that it produces an implicit representation of the set of all PPH solutions. In HAP, that representation is used to enumerate all the PPH solutions (for the rows not removed) in order to choose one that best conforms to the assumption that the observed genotypes were created by random mating of the inferred haplotypes, and therefore fits the Hardy-Weinberg equilibrium.

In addition to the assumption of random mating, the implicit model of haplotype evolution embodied in HAP is that the rare haplotypes are created by recent recombinations of a few common haplotypes, or by recent mutations of common haplotypes. That view of haplotype evolution is articulated in [23, 65]. The near-perfect phylogeny model goes one step further, by asserting that the common haplotypes fit a perfect-phylogeny model.

We consider the observation in [42] that well-studied data nearly fit the perfect phylogeny model to be a validation of the original PPH idea. The strict PPH model may be overly-idealized, or too brittle to handle errors in real data, but it is valuable to have a precisely-specified model that leads to an efficiently-solved computational problem that can be used in the core of other more heuristic programs (such as HAP) to handle more complex (and messier) real-world data. An initial effort to more formally specify a model of imperfect phylogeny haplotyping, and to solve the associated computational problems, was recently published in [70].

1.6 Population-Genetic and Statistical Methods

Much of the work described above has been carried out by computer scientists and/or with the methods used by computer scientists. A number of other important approaches to the problem of haplotype inference have originated mainly in the research community of population geneticists and statisticians interested in the assessment and analysis of genetic variation.

Of central historical and scientific note in this regard is the use of maximum likelihood to estimate haplotype frequencies and to infer haplotype pairs. It is straightforward to write down the likelihood function associated with any given sample of individuals if one makes an assumption about the process by which mating occurs within the population. The standard and usually reasonable assumption is that there is a process of random mating among individuals. Given this assumption, one can then derive an explicit likelihood function and the goal is to determine its maximum value [75]. This value will yield estimates of the haplotype frequencies underlying the observed genotype frequencies. Given the haplotype frequencies, one can then determine the most probable pair of haplotypes that underlies any given ambiguous genotype. The main problem in practice then is the derivation of the maximum value of the likelihood function.

For two SNP sites, one can analytically derive the maximum value [64]. This fact has long been known but unfortunately, this approach has almost never been exploited as a means of solution for this important case (exceptions are [72, 78]). Instead, for this case and for the m-site case, the method of expectation-maximization (EM) has been used [27, 44, 60, 66]. The use of this numerical approach to determine the maximum value of the likelihood function has both positive and negative consequences.

On the one hand, it allows the estimation of haplotype frequencies for data sets for which there are few, if any, alternative estimation approaches available. The importance of this can hardly be overestimated. On the other hand, a numerical method such as EM yields only limited information about the overall "dimensionality" of the estimation problem. So, for example, typical use of the EM algorithm does not reveal whether there are multiple peaks on the likelihood surface or whether the surface around a peak is flat or steep. Such information has obvious important implications for the confidence one has in any given result of the application of EM. Examples are shown in Orzack *et al.* [64] of two-site genotype data sets for which the EM algorithm yields misleading results.

In the case of two sites, we recommend use of the analytical method presented in [64]. For data sets with more than two sites, the EM approach is certainly one that should be considered, if one uses it in such a way that the topography of the likelihood surface is at least partially described. The simple way of doing this is by starting the algorithm with many different initial estimates of the unknown haplotype frequencies. If all such different estimates result in the same value of the likelihood function then one can have more confidence that at least there is just one peak. It is not clear with this (or any other method, see below) how to proceed if one has multiple peaks, but at least knowing of their existence is better than proceeding in ignorance.

The likelihood approach can be viewed as a special case of a Bayesian-inference problem. In this case, one has a flat prior, implying that no one prior estimate of haplotype frequencies is better than any other. In the last five years, a number of alternative approaches have been developed in which more informative prior information is incorporated into the estimation procedure, so as to get better estimates of the unknown haplotype frequencies and/or haplotype pairs. Of note in this regard are the programs called Phase [69] and Haplotyper [61]. In the case of Phase, the Bayesian-prior is derived from the infinite-sites model and to this extent, it is a plausible prior to use for many (but not all) data sets. The Gibbs sampler is then used to calculate the posterior distribution from which one can derive estimates of haplotype frequencies and infer haplotype pairs. Further elaboration and discussion of this approach can be found in [58, 68].

In contrast, the other Bayesian method, Haplotyper, uses a prior that is not derived from an explicit population-genetic model; it is consistent with a model of inheritance in which sequences of parents and offspring are potentially independent of one another [68]. The Dirichlet distribution is used here as the sampling distribution. This prior is clearly less biologically meaningful than the prior used in Phase. The Gibbs sampler is also used to calculate the posterior distribution.

What all of these methods have in common is the use of a numerical method to derive estimates of haplotype frequencies and predictions of haplotype pairs. In addition, all of these calculations are stochastic in the sense that one must start each execution with different initial haplotype frequencies (in the cases of Phase and Haplotyper) or should do so (in the case of EM). To this extent, the concern is that the reliability of results derived from any one execution is uncertain. This problem has been recognized by the creators of some of these programs (e.g, [69]) but the resulting implications for how these programs should be used have not been adequately explored. So, typical analyses based on these methods have involved a single execution of a program. How meaningful the associated results are is very unclear; at the very least, it is easy to find genotypic configurations for which different executions of any given method can result in very different estimates of haplotype frequencies (see [64]).

The main problem here is not the discovery of alternative solutions for any given data set. The only requirement is sufficient computing power. Several thousand executions of even the most time-intensive methods that are presently available can be achieved in a few days on the fastest available microprocessors. Accordingly, the central and unresolved problem is what one can make of the possible multiple solutions that one may find for any given data set. One possibility is the use of consensus, just as it was applied in the analysis of multiple solutions stemming from the rule-based algorithms such as Clark's method (as described in Section 1.3.4). Consensus has also been applied in [29]. It is clear that there is great promise to the consensus approach and it may prove widely useful. However, this remains to be seen and additional applications and theoretical analysis are needed. Of course, in any given instance, multiple executions of an inference program may result in the same solution. Such was the case for several thousand Phase analyses of the APOE data set described above (e.g., [63]). However, how typical such monomorphism is remains unknown. Given present evidence and the black-box nature of present stochastic inference methods, we strongly caution against an expectation that the results of any method should be regarded as "true" even if multiple executions result in the same solution. Such confidence can come only from improved understanding of the performance of the algorithms and especially from analyses in which the accuracy of any given method is assessed by a comparison of inferred and real haplotype pairs.

1.7 Going Forward

Our hope is that this review provides a meaningful and stimulating assessment of the present state of the biologically important problem of haplotype inference. While important progress has been made, it is clear that there are substantial questions and issues that remain to be resolved. We hope and expect that further progress will come from the separate and combined efforts of biologists, computer scientists, and statisticians. The interdisciplinary nature of this research effort is testimony to the remarkable state of present research in bioinformatics.

1.8 Acknowledgments

Research partially supported by award EIA-0220154 from the National Science Foundation and award P01-AG0225000-01 from the National Institute of Aging.

References

[1] www.hapmap.org.

[2] R. M. Adkins. Comparison of the accuracy of methods of computational haplotype inference using a large empirical dataset. *BMC Genetics*, 5(22), 2004.

[3] D. Altshuler and A. Clark. Harvesting medical information from the Human family tree. *Science*, 307:1052–1053, 2005.

[4] V. Bafna, D. Gusfield, S. Hannenhalli, and S. Yooseph. A note on efficient computation of haplotypes via perfect phylogeny. *Journal of Computational Biology*, 11(5):858-866, 2004.

[5] V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph. Haplotyping as perfect phylogeny: A direct approach. *Journal of Computational Biology*, 10:323–340, 2003.

[6] T. Barzuza, J.S. Beckman, R. Shamir, and I. Pe'er. Computational Problems in perfect phylogeny haplotyping: XOR genotypes and TAG SNPs In *Thirteenth Annual Symposium on Combinatorial Pattern Matching (CPM'04)*, p. 14-31, 2004.

[7] T. Barzuza and I. Pe'er. personal communication.

[8] R. E. Bixby and D. K. Wagner. An almost linear-time algorithm for graph realization. *Mathematics of Operations Research*, 13:99–123, 1988.

[9] P. Bonizzoni, G. Della Vedova, R. Dondi, and J. Li. The haplotyping problem: Models and solutions. *Journal of Computer Science and Technology*, 18:675–688, 2003.

[10] D. Brown and I. Harrower. A new integer programming formulation for the pure parsimony problem in haplotype analysis *Proceedings of the 2004 Workshop on Algorithms in Bioinformatics* Springer Lecture Notes in Bioinformatics, LNCS, Vol. 3240 p. 254-265

[11] D. Brown and I. Harrower. A new formulation for haplotype inference by pure parsimony. Technical report, University of Waterloo, School of Computer Science. report CS-2005-03.

[12] R.H. Chung and D. Gusfield. Empirical exploration of perfect phylogeny haplotyping and haplotypers. *Proceedings of the 9th International Conference on Computing and Combinatorics COCOON03.* Spring Lecture Notes in Computer Sciencen, Vol. 2697, pages 5–19, 2003.

[13] R.H. Chung and D. Gusfield. Perfect phylogeny haplotyper: Haplotype inferral using a tree model. *Bioinformatics*, 19(6):780–781, 2003.

[14] A. Clark. Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular Biology and Evolution*, 7:111–122, 1990.

[15] A. Clark, K. Weiss, D. Nickerson, et al. Haplotype structure and population genetic inferences from nucleotide-sequence variation in human lipoprotein lipase. *American Journal of Human Genetics*, 63:595–612, 1998.

[16] A. G. Clark. Finding genes underlying risk of complex disease by linkage disequilibrium mapping. *Current Opinion in Genetics & Development*, 13:296–302, 2003.

[17] International HapMap Consortium. HapMap project. *Nature*, 426:789–796, 2003.

[18] M. Daly, J. Rioux, S. Schaffner, T. Hudson, and E. Lander. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29:229–232, 2001.

[19] P. Damaschke. Fast perfect phylogeny haplotype inference. *14th Symposium on Fundamentals of Computation Theory FCT*, 2751:183–194, 2003.

[20] P. Damaschke. Incremental haplotype inference, phylogeny and almost bipartite graphs. *2nd RECOMB Satellite Workshop on Computational Methods for SNPs and Haplotypes*, pages 1–11, 2004.

[21] Z. Ding, V. Filkov, and D. Gusfield. A linear-time algorithm for the perfect phylogeny haplotyping problem. *Proceedings of the Ninth Annual International Conference on Computational Biology (RECOMB 2005).* S. Miyano, J. Mesirov, S. Kasif, S. Istrail, P. Pevzner, and M. Waterman (eds). Springer Lecture Notes in Bioinformatics, LNCS Vol. 3500 p. 585-600.

[22] P. Donnelly. Comments made in a lecture given at the DIMACS conference on Computational Methods for SNPs and Haplotype Inference, November 2002.

[23] N. El-Mabrouk and D. Labuda. Haplotype histories as pathways of recombinations. *Bioinformatics*, 20:1836–1841, 2004.

[24] E. Eskin, E. Halperin, and R. M. Karp. Large scale reconstruction of haplotypes from genotype data. *Proceedings of the 7th Annual International Conference on Computational Biology (RECOMB 2003).* p. 104-113

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

[26] E. Eskin, E. Halperin, and R. Sharan. Optimally phasing long genomic regions using local haplotype predictions. In *Proceedings of the Second RECOMB Satellite Workshop on Computational Methods for SNPs and Haplotypes*, February 2004. Pittsburgh, USA.

[27] L. Excoffier and M. Slatkin. Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Molecular Biology and Evolution*, 12:921–927, 1995.

[28] S. M. Fullerton, A. Clark, and Charles Sing et. al. Apolipoprotein E variation at the sequence haplotype level: implications for the origin and maintenance of a major human polymorphism. *American Journal of Human Genetics*, 67:881–900, 2000.

[29] S. M. Fullerton, A. V. Buchanan, V. A. Sonpar, S. L. Taylor, J. D. Smith, C. S. Carlson, V. Salomaa, J. H. Stengard, E. Boerwinkle, A. G. Clark, D. A. Nickerson, and K. M. Weiss. The effects of scale: variation in the apoa1/c3/a4/a5 gene cluster. *Human Genetics*, 115:36–56, 2004.

[30] F. Gavril and R. Tamari. An algorithm for constructing edge-trees from hypergraphs. *Networks*, 13:377–388, 1983.

[31] J. Gramm, T. Nierhoff, R. Sharan, and T. Tantau. On the complexity of haplotyping via perfect phylogeny. In *Second RECOMB Satellite Workshop on Computational Methods for SNPs and Haplotypes*, Pittsburgh, USA, February 2004. In Springer Lecture Notes in Bioinformatics, LNCS 2004.

[32] J. Gramm, T. Nierhoff, and T. Tantau. Perfect path phylogeny haplotyping with missing data is fixed-parameter tractable. In *First International Workshop on Parametrized and Exact Computation (IWPEC 2004)*, Bergen, Norway, September 2004. In LNCS, Springer.

[33] D. Gusfield. Efficient algorithms for inferring evolutionary history. *Networks*, 21:19–28, 1991.

[34] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.* Cambridge University Press, Cambridge, UK, 1997.

[35] D. Gusfield. A practical algorithm for deducing haplotypes in diploid populations. In *Proceedings of 8th International Conference on Intelligent Systems in Molecular Biology*, pages 183–189. AAAI Press, 2000.

[36] D. Gusfield. Inference of haplotypes from samples of diploid populations: complexity and algorithms. *Journal of Computational Biology*, 8(3):305-323, 2001.

[37] D. Gusfield. Haplotyping as Perfect Phylogeny: Conceptual Framework and Efficient Solutions. In *Proceedings of RECOMB 2002: The Sixth Annual International Conference on Computational Biology*, pages 166–175, 2002. Extended Abstract.

[38] D. Gusfield. Haplotype inference by pure parsimony. In E. Chavez R. Baeza-Yates and M. Crochemore, editors, *14th Annual Symposium on Combinatorial Pattern Matching (CPM'03)*, volume 2676, pages 144–155. Springer LNCS, 2003.

[39] D. Gusfield. An overview of combinatorial methods for haplotype inference. In S. Istrail, M. Waterman, and A. Clark, editors, *Computational Methods for SNPs and Haplotype Inference*, volume 2983, pages 9–25. Springer, 2004. Lecture Notes in Computer Science.

[40] B. Halldorsson, V. Bafna, N. Edwards, R. Lipert, S. Yooseph, and S. Istrail. Combinatorial problems arising in SNP and haplotype analysis. In C. Calude, M. Dinneen, and V. Vajnovski, editors, *Discrete Mathematics and Theoretical Computer Science. Proceedings of DMTCS 2003*, volume 2731, pages 26–47. Springer, 2003. Springer

Lecture Notes in Computer Science.

[41] B. Halldorsson, V. Bafna, N. Edwards, R. Lipert, S. Yooseph, and S. Istrail. A survey of computational methods for determining haplotypes. In *Proceedings of the First RECOMB Satellite on Computational Methods for SNPs and Haplotype Inference.* Springer Lecture Notes in Bioinformatics, LNCS, Vol. 2983 p. 26-47, 2003.

[42] E. Halperin and E. Eskin. Haplotype reconstruction from genotype data using imperfect phylogeny. *Bioinformatics*, 20:1842–1849, 2004.

[43] E. Halperin and R. Karp. Perfect phylogeny and haplotype assignment. *Proceedings of The 8th Ann. International Conference Research in Computational Molecular Biology (RECOMB 2004*, pages 10–19. ACM Press, 2004.

[44] M. Hawley and K. Kidd. HAPLO: a program using the EM algorithm to estimate the frequencies of multi-site haplotypes. *Journal of Heredity*, 86:409–411, 1995.

[45] J. He and A. Zelikovsky. Linear reduction for haplotype inference. In *Proc. of 2004 Workshop on Algorithms in Bioinformatics*, Springer Lecture Notes in Bioinformatics, LNCS Vol. 3240. pages 242-253.

[46] L. Helmuth. Genome research: Map of the human genome 3.0. *Science*, 293:583–585, 2001.

[47] D. Hinds, L. Stuve, G. Nilsen, E. Halperin, E. Eskin, D. Gallinger, K. Frazer, and D. Cox. Whole-genome patterns of common DNA variation in three human populations. *Science*, 307:1072–1079, 2005.

[48] Y.T. Huang, K.M. Chao, and T. Chen. An approximation algorithm for haplotype inference by maximum parsimony. *ACM Symposium for Applied Computing SAC'05*, 2005.

[49] E. Hubbell. Personal communication. August 2000.

[50] R. Hudson. Gene genealogies and the coalescent process. *Oxford Survey of Evolutionary Biology*, 7:1–44, 1990.

[51] S. Iwata. University of Tokyo. Personal Communication.

[52] G. Kimmel, and R. Shamir. The Incomplete Perfect Phylogeny Haplotype Problem. *J. Bioinformatics and Computational Biology.* Vol. 3:1-25, 2005

[53] P.Y. Kwok. Genomics: Genetic association by whole-genome analysis? *Science*, 294:1669–1670, 2001.

[54] G. Lancia, C. Pinotti, and R. Rizzi. Haplotyping populations: Complexity and approximations. Technical Report dit-02-082, University of Trento, 2002.

[55] G. Lancia, C. Pinotti, and R. Rizzi. Haplotyping populations by pure parsimony: Complexity, exact and approximation algorithms. *INFORMS Journal on Computing, special issue on Computational Biology*, 16:348–359, 2004.

[56] J. Lee. U.T. Austin. Personal Communication.

[57] Z. Li and W. Zhou and X. Zhang and L. Chen. A Parsimonious tree-grow method for haplotype inference. *Bioinformatics.* 2005, 21:3475-3481.

[58] S. Lin, D. Cutler, M. Zwick, and A. Chakravarti. Haplotype inference in random population samples. *American Journal of Human Genetics*, 71:1129–1137, 2003.

[59] L. Löfgren. Irredundant and redundant boolean branch networks. *IRE Transactions on Circuit Theory*, CT-6:158–175, 1959.

[60] J.C. Long, R.C. William, and M. Urbanek. An E-M algorithm and testing strategy for multiple-locus haplotypes. *American Journal of Human Genetics*, 56:799–810, 1995.

[61] T. Niu, Z. Qin, X. Xu, and J.S. Liu. Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *American Journal of Human Genetics*, 70:157–169, 2002.

[62] T. Ohto. An experimental analysis of the Bixby-Wagner algorithm for graph realization problems. *IPSJ SIGNotes ALgorithms Abstract,*

http://www.ipsj.or.jp/members/SIGNotes/Eng/16/2002/084/article001.html, 084-001, 2002.

[63] S.H. Orzack, D. Gusfield, J. Olson, S. Nesbitt, L. Subrahmanyan, and V. P. Stanton Jr. Analysis and exploration of the use of rule-based algorithms and consensus methods for the inferral of haplotypes. *Genetics*, 165:915–928, 2003.

[64] S.H. Orzack, L. Subrahmanyan, D. Gusfield, S. Lissargue, and L. Essioux. A comparison of an exact method and algorithmic method for haplotype frequency inferral. Preprint, 2005.

[65] D. Posada and K. Crandall. Intraspecific gene genealogies: trees grafting into networks. *Trends in Ecology and Evolution*, 16:37–45, 2001.

[66] Z. Qin, T. Niu, and J.S. Liu. Partition-ligation-expectation-maximization algorithm for haplotype inference with single-nucleotide polymorphisms. *American Journal of Human Genetics*, 71:1242–1247, 2002.

[67] R. V. Satya and A. Mukherjee. An Optimal Algorithm for Perfect Phylogeny Haplotyping. *Proceedings of 4th CSB Bioinformatics Conference.* IEEE Press, Los Alamitos, CA, 2005.

[68] M. Stephens and P. Donnelly. A comparison of Bayesian methods for haplotype reconstruction from population genotype data. *American Journal of Human Genetics*, 73:1162–1169, 2003.

[69] M. Stephens, N. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction from population data. *American Journal of Human Genetics*, 68:978–989, 2001.

[70] Y. Song, Y. Wu and D. Gusfield. Haplotyping with one homoplasy or recombination event. Proceedings of Workshop on Algorithms in Bioinformatics (WABI) 2005. Springer, Lecture Notes in Bioinformatics, LNCS Vol. 3692.

[71] S. Tavare. Calibrating the clock: Using stochastic processes to measure the rate of evolution. In E. Lander and M. Waterman, editors, *Calculating the Secrets of Life.* National Academy Press, 1995.

[72] De Vivo, I., G. S. Huggins, S. E. Hankinson, P. J. Lescault, M. Boezen, G. A. Colditz, and D. J. Hunter. A functional polymorphism in the promoter of the progesterone receptor gene associated with endometrial cancer risk. *Proceedings of the National Academy of Science (USA)*, 99:12263–12268, 2002.

[73] J. D. Wall and J. K. Pritchard. Haplotype blocks and linkage disequilibrium in the human genome. *Nature Reviews*, 4:587–597, 2003.

[74] L. Wang and L. Xu. Haplotype inference by maximum parsimony. *Bioinformatics*, 19:1773–1780, 2003.

[75] B. S. Weir. *Genetic Data Analysis II* Sinauer Associates 1996

[76] W. T. Whitney. 2-isomorphic graphs. *American Mathematics Journal*, 55:245–254, 1933.

[77] C. Wiuf. Inference of recombination and block structure using unphased data. *Genetics*, 166:537–545, 2004.

[78] R. Y. L. Zee, H. H. Hegener, N. R. Cook, and P. M. Ridker. C-reactive protein gene polymorphisms and the risk of venous thromboembolism: a haplotype-based analysis. *Journal of Thrombosis and Haemostasis*, 2:1240–1243, 2004.

[79] J. Zhang, W. Rowe, A. Clark, and K. Buetow. Genomewide distribution of high-frequency, completely mismatching SNP haplotype pairs observed to be common across Human populations. *American Journal of Human Genetics*, 73:1073–1081, 2003.

Index

ambiguous site, 1-3
APOE, 1-6
association mapping, 1-13

Bayesian-inference, 1-20
branch-and-bound method, 1-10

Clark's algorithm, 1-3
  genetic model, 1-5
  inference rule, 1-3
  maximum resolution, 1-4
  MR problem, 1-5
  multiple solutions, 1-6
  stochastic behavior, 1-6
coalescent, 1-13
combinatorial methods, 1-3
consecutive ones problem, 1-17
consensus solution, 1-6, 1-21

diploid organism, 1-2
Dirichlet distribution, 1-20

Expectation-Maximization (EM), 1-20

genetic model, 1-3, 1-10, 1-18
  Clark's algorithm, 1-5
  coalescent, 1-12
  Hardy-Weinberg, 1-19
  infinite sites, 1-5, 1-12, 1-20
  perfect phylogeny, 1-12, 1-14, 1-19
  prior distribution, 1-20
  pure parsimony, 1-7
  random mating, 1-19
genotype, 1-1, 1-2
Gibbs sampler, 1-20
graph-realization, 1-13, 1-14, 1-17

HAP, 1-18
haplotype, 1-1, 1-2
  common, 1-5, 1-18, 1-19
  evolution, 1-3
  inference, 1-1, 1-3
  infrequent, 1-18
haplotype block, 1-17, 1-18
  dispersed, 1-14
haplotype map, 1-1
Haplotyper, 1-20
Hardy-Weinberg equilibrium, 1-19

inference rule, 1-3
infinite sites model, 1-5, 1-12
integer linear programming, 1-5, 1-7, 1-8
  hybrid formulation, 1-12
  polynomial size formulation, 1-10
  RTIP, 1-9
  TIP, 1-9
integer quadratic programming, 1-12

Löfgren's method, 1-13, 1-15
LD, 1-14, 1-17
linear algebra, 1-16
linkage disequilibrium (LD), 1-14

maximum likelihood, 1-19
  two sites, 1-20
multiple solutions, 1-6, 1-20, 1-21

near-perfect phylogeny haplotyping, 1-18
  HAP, 1-18

parsimony criterion, 1-7
perfect phylogeny, 1-12
perfect phylogeny haplotyping, 1-12, 1-13
  boolean matrix multiplication, 1-16
  incomplete data, 1-16
  linear time algorithm, 1-14
  near-perfect phylogeny, 1-18
  phase transition, 1-15
  related results, 1-16
  uniqueness, 1-15, 1-17
  XOR problem, 1-17
  yin-yang haplotypes, 1-17
Phase, 1-7, 1-10, 1-20, 1-21
PPH programs, 1-13
Pure-Parsimony problem, 1-7, 1-10

recombination, 1-9, 1-10, 1-12–1-14
resolved site, 1-3
rule-based methods, 1-6

sequence variation, 1-2
shadow tree, 1-15
SNP, 1-1, 1-2
  tag, 1-18

statistical methods, 1-3, 1-19

tag genotype SNP, 1-18
tag SNP, 1-18
three-partition, 1-15

Whitney's theorem, 1-15

XOR PPH problem, 1-17