Large Scale Reconstruction of Haplotypes from Genotype Data

Eleazar Eskin
Computer Science Department
Columbia University
E-mail: eeskin@cs.columbia.edu

Eran Halperin
CS Division
Soda Hall
University of California Berkeley
CA 94720-1776
E-mail: eran@eecs.berkeley.edu

Richard M. Karp
International Computer Science Institute
1947 Center St.
Berkeley, CA 94704
E-mail:karp@cs.berkeley.edu

ABSTRACT

Critical to the understanding of the genetic basis for complex diseases is the modeling of human variation. Most of this variation can be characterized by single nucleotide polymorphisms (SNPs) which are mutations at a single nucleotide position. To characterize an individual's variation, we must determine an individual's *haplotype* or which nucleotide base occurs at each position of these common SNPs for each chromosome. In this paper, we present results for a highly accurate method for haplotype resolution from genotype data. Our method leverages a new insight into the underlying structure of haplotypes which shows that SNPs are organized in highly correlated "blocks". The majority of individuals have one of about four common haplotypes in each block. Our method partitions the SNPs into blocks and for each block, we predict the common haplotypes and each individual's haplotype. We evaluate our method over biological data. Our method predicts the common haplotypes perfectly and has a very low error rate (0.47%) when taking into account the predictions for the uncommon haplotypes. Our method is extremely efficient compared to previous methods, (a matter of seconds where previous methods needed hours). Its efficiency allows us to find the block partition of the haplotypes, to cope with missing data and to work with large data sets such as genotypes for thousands of SNPs for hundreds of individuals. The algorithm is available via webserver at `http://www.cs.columbia.edu/compbio/hap/` .

Categories and Subject Descriptors

J.3 [**Computer Applications**]: Life and Medical Sciences–biology and genetics

General Terms

Algorithms, Theory

Keywords

Haplotype Resolution, Single Nucleotide Polymorphisms (SNPs), Perfect Phylogeny

1. INTRODUCTION

Critical to the understanding of the genetic basis for complex diseases is the modeling of human variation. Most of this variation can be characterized by single nucleotide polymorphisms (SNPs) which are mutations at a single nucleotide position that occurred once in human history and were passed on through heredity. Approximately 10 million common SNPs[19, 10], each with a frequency of 10% to 50% account for the majority of the variation between DNA sequences of individuals[22]. To characterize an individual's variation, we must determine an individual's *haplotype* or which nucleotide base occurs at each position of these common SNPs for each chromosome. By correlating an individual's haplotypes with the presence of a disease, researchers can better understand complex diseases. The effort to characterize human variation, currently a major focus for the NIH, will be a tremendous undertaking requiring obtaining the haplotype information from a large collection of individuals from diverse populations [19].

Although the two chromosomes of an individual can be separated and analyzed independently as in Patil et al., 2001 [22], current technology suitable for large scale polymorphism screening obtains *genotype* information at each SNP. The genotype gives the bases at each SNP for both copies of the chromosome, but loses the information as to the chromosome on which each base appears. Consider a SNP where there are two common bases, A or G. There are four possible cases for the haplotype. Two of the cases are where either both chromosomes contains A or both chromosomes contains G. We refer to these cases as *homozygous* genotypes. The other two cases are where the first chromosome contain A and the second contain G and vice versa. We refer to these cases as *heterozygous* genotypes. For this SNP,

Haplotype	0,1 Representation	Frequency
CCGAT	00000	66
CTGAC	01001	24
ATACT	11110	10
CTGAT	01000	6
ATGAT	11000	1
ATGCC	11011	1
CCGAC	00001	1

Table 1: Block 6 from Daly et al. 2001, [5]. The block contains 5 SNPs over 11 kilobases. The horizontal line separates the common haplotypes from rare haplotypes. The first column shows the haplotypes from the transmitted chromosomes. The second column shows the same haplotypes but mapped to 0,1 representation. The 0 represents the common nucleotide at the position, while the 1 represents the rare nucleotide at the position. The third column is the frequency of the haplotype block in the transmitted chromosomes. Note that any chromosome that contained any ambiguity in the block due either to missing data or heterozygous genotypes for all members of the trio was omitted.

there are three possible cases for the genotype information. In the homozygous cases, the genotype will be either A or G respectively and we can infer that the base appears in both chromosomes. In the heterozygous cases, the genotype will be H (for heterozygous) and we can infer that in one chromosome, we have an A and in the other we have a G, but we can not infer on which chromosome each appears. This causes problems in reconstructing the haplotypes. Consider the example where an individual at four successive SNPs, with possible values A or G, has a genotype $AHHG$. In this case, the individual's haplotypes have two possibilities: either $AAAG$ on one chromosome and $AGGG$ on the other chromosome or $AAGG$ and $AGAG$. Without any other information, such as the genotypes from related individuals, it is impossible to determine the individual's actual haplotypes. This problem of haplotype resolution is often referred to as the phase problem.

Consider a set of n neighboring SNPs. There are at most 4^n possible haplotypes that occur for each block. Since there are only two observed bases for the vast majority of SNPs, there are at most 2^n haplotypes. Recent studies in linkage disequilibrium [9, 23] characterizing haplotype structure have found that in fact there are far fewer haplotypes. These studies have shown that SNPs are grouped into "blocks" of limited diversity with the regions between blocks being "hot spots" of recombination. In each block containing n SNPs, typically around four haplotypes account for the majority of the haplotypes in the population. Consider the haplotype block shown in Table 1 consisting of 5 SNPs over 11 kilobases from a recent paper, Daly et al, 2001. Note that 90% of the individuals contain one of four common haplotypes.

Haplotypes can be resolved from genotype data by making the assumption that most of the haplotypes within a block will loosely fit the perfect phylogeny model. This method for resolving haplotypes was first proposed in Gusfield, 2002 [13]. The perfect phylogeny model assumes an infinite site mutation model and allows no recombinations [15]. The infinite site mutation model makes the assumption that at each SNP site, a mutation only happened once in human history. This model forbids recurrent mutations or back mutations. The assumptions of the model imply that a chromosome with a mutation at a SNP is a direct descendant from the chromosome of the ancestor in which the mutation occurred. Likewise, any chromosome without the mutation can not be a descendant of a chromosome that has the mutation. Clearly, these assumptions are not realistic although it is reasonable to assume that recombinations and recurrent mutations are relatively rare events within a block. Thus, we consider a relaxed model which allows for a certain number of recurrent mutations and recombinations within a block.

In this paper, we present results for a highly accurate method for haplotype resolution from genotype data. Our method takes as input a population of genotypes and decomposes the SNPs into blocks. For each block we predict the common haplotypes as well as the haplotypes of each individual in the population. We also show that the common haplotypes roughly fit a perfect phylogeny model.

We evaluate our method over the data collected in the study of a 500 kilobase region of chromosome 5p31 containing 103 SNPs. In this study, genotypes are collected from 129 mother, father, child trios where the correct child haplotypes are inferred from these genotypes. Using genotypes for all members of a mother, father, child trio, we can infer the haplotypes in most cases using Mendelian genetics. We know the child inherited one chromosome from the mother and one from the father assuming there was no recombination. Consider the example above where again in four successive SNPs where the possible bases are A and G, The child's genotype is $AHHG$ as above, the mother's genotype is $HHGH$ and the father's genotype is $HAHG$. In this case we can infer the haplotypes of all members of the trio. Each parent has one *transmitted* chromosome and one *untransmitted* chromosome. The child's haplotypes consist of the transmitted chromosomes from both the mother and father. Since in the child, the genotype at the first SNP is A while in both parents the genotypes are heterozygous, we can infer that in both the mother's and father's transmitted chromosomes, the SNP is A, while in the untransmitted chromosomes the SNP is G. At the second SNP, since the father's genotype is homozygous A, we know that both the father's transmitted and untransmitted chromosomes must contain A, while the mother's transmitted chromosome must contain the base G in order for the child's genotype to be heterozygous. If we continue this analysis, we can infer that the child's haplotypes are $AAAG$ and $AGGG$ while the mother's transmitted chromosome is $AGGG$ and untransmitted chromosome is $GAGA$ and the father's transmitted chromosome is $AAAG$ and untransmitted chromosome is $GAGG$. Using this dataset, we can verify the effectiveness of our method. Using only the population of the children's genotypes, we make prediction for their haplotypes. We can verify our predictions by inferring the haplotype for each child using the genotypes of the child's parents. Essentially, our method can effectively predict the haplotypes for *unrelated* individuals. This ability significantly reduces the costs and difficulties of characterizing human variation since it eliminates the need for collecting genotype data from complete trios.

In Daly et al., 2001 [5], the 103 SNPs were split into 11 blocks of varying lengths and number of SNPs contained in each block. Over these blocks, our method very accurately predicts the haplotypes from the genotypes. In all cases but one, we predict correctly all of the common haplotypes for each block. The predictions for the individuals' haplotypes differed from the actual haplotypes by less than 0.30% in terms of bases. Our errors occur only in the individuals that have a haplotype that is uncommon. In addition, approximately 10% of the genotype data is missing due to experimental error. We make predictions for the haplotypes corresponding to the missing genotype data also with less than 0.53% error. The method makes its prediction for each block in mere seconds, significantly faster than other methods for haplotype resolution.

Since a priori, we do not know the block partitions, the method also predicts block partitions from genotype data using the criteria of minimizing the number of "representative SNPs" as presented in Patil et al., 2001 [22]. Over the blocks chosen by the algorithm, the error rate for the 103 SNPs is 0.73% and only 0.47% if we ignore the missing data. In practice, there are multiple possible block partitions where the majority of the individuals contain relatively few haplotypes. Since these block predictions overlap, we are able to make overlapping predictions for local haplotypes and use a "tiling" technique to reconstruct the entire haplotypes for each individual.

We show that our method in addition to being highly accurate can scale to very large samples. We consider a sample consisting of 24,047 SNPs from Patil et al., 2001 [22]. We generate a set of genotype data using the haplotypes from this data creating an artificial population of several hundred individuals. We then show that our algorithm can scale to this sample size and predict the haplotypes and partition them into blocks.

The core of the method is an algorithm, described in detail in Eskin et al., 2002 [6], that determines all possible haplotype resolutions that are consistent with a perfect phylogeny model. This approach was first proposed in Gusfield, 2002 [13], where a polynomial time algorithm was introduced to the problem. The algorithm in Gusfield, 2002 [13] uses a complicated but elegant machinery from matroid theory, and its asymptotic performance is near optimal. However, the algorithm in Eskin et al., 2002 [6] is much simpler and easier to implement and as we show in this paper, more easily extended to handle cases where the data is not consistent with a perfect phylogeny model. These extensions are critical to the application of the algorithm because as in real data [5], the haplotypes only roughly fit the perfect phylogeny model. This algorithm is used to determine a set of candidate haplotype resolutions for each block. We then use a maximum likelihood model to choose the best haplotype resolution from this set of candidates. In practice, about 10% of the genotype data is missing and the maximum likelihood model correctly resolves the vast majority of this missing data. We note that independently of our work, Bafna et. al, 2002 [2] found and implemented a simpler algorithm then the one given in Gusfield, 2002 [13], with the same asymptotic performance as our algorithm.

Existing methods effectively assume all of the SNPs are in a single block. These methods include a variety of methods including the parsimony approach of Clark [3] and related approaches [11, 12, 16], maximum likelihood methods [7, 8, 14, 17] and statistical methods such as PHASE [26], and perfect phylogeny-based approaches [13]. All of these approaches suffer from the fact that they do not explicitly take into account the haplotype block structure. In addition, these methods are often too inefficient to be practical for large data sets. These methods typically can not scale to data that contains more than 30 sites. A similar approach to ours is the strict perfect phylogeny model approach of Gusfield, 2002 [13]. However, in this paper, we show that only if we relax the assumptions of the perfect phylogeny model does the algorithm work in practice.

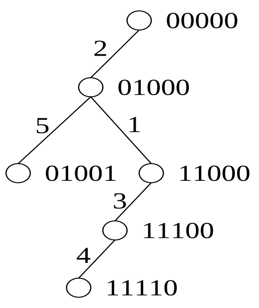

Figure 1: The perfect phylogeny tree corresponding to the haplotypes 00000, 01000, 01001, 11000, 11100, 11110**.**

2. PRELIMINARIES

We first formally describe the perfect phylogeny model and the PPH problem. We assume that at each polymorphic site there are two possible nucleotides that appear at any position in any one of the chromosomes. Let us denote these nucleotides by 0 and 1. We note that although the assumption that there are only two possibilities at any site seems artificial, it is the case in most polymorphic sites.

2.1 The Perfect Phylogeny Haplotype Problem

Given a $(0, 1)$- matrix $B = (b_{ij})$ of size $2n \times m$ which represents a set of haplotypes, we say that B fits the perfect phylogeny model if there exists a rooted tree $T(B)$ representing the evolution of the haplotypes such that the following holds:

1. Each vertex v of $T(B)$ is labeled by a row vector $l(v)$ of length m representing a possible haplotype. Each coordinate position in these row vectors corresponds to a site.
2. For each coordinate i , there is at most one pair (u, v) of vertices such that u is the parent of v and $l(u)$ differs from $l(v)$ in coordinate i..
3. The set of rows of B is contained in the set of labels of $T(B)$.

An example of a perfect phylogeny tree is given in Figure 1. Throughout the paper, for a given integer k, we denote the set $\{1, \ldots, k\}$ by $[k]$.

Let a be a m-vector whose elements are drawn from $\{0, 1, 2\}$. Let b and c be m-vectors whose elements are drawn from $\{0, 1\}$. The vectors b and c are *compatible* with a if, for each coordinate i the following hold:

1. if $b(i) = c(i)$ then $a(i) = b(i) = c(i)$;
2. if $b(i) \neq c(i)$ then $a(i) = 2$.

Thus, if genotype a is derived from haplotypes b and c, then b and c must be compatible with a.

A $n \times m$ matrix A with elements drawn from $\{0, 1, 2\}$ is *realizable* if there exists a $(0, 1)$-matrix B of size $2n \times m$ that fits the perfect phylogeny model such that, for every row a of A, B contains a pair of rows compatible with a. In such a case the matrix B is called a *legal extension* of A.

The Perfect Phylogeny Haplotype Problem (PPH). Given a $(0, 1, 2)$- matrix A, either find a legal extension of A or determine that A is not realizable.

The problem has the following interpretation. The matrix A represents the genotypes of n individuals, where the length of each genotype is m. For any individual $r \in [n]$, and site $c \in [m]$, the four possible haplotype states are $(0,0)$,$(0,1)$,$(1,0)$, and $(1,1)$. The sequence observed can only distinguish between the three states $0, 1$ and 2. States 0 and 1 stand for the haplotype states $(0,0)$, $(1,1)$ respectively. State 2 stands for either the haplotype state $(0,1)$ or $(1,0)$. We are interested in constructing the haplotype matrix B which will be consistent with A and with the perfect phylogeny model.

2.2 Some Useful Lemmas and Notations

Throughout the paper we will use the terms *site* and *column* interchangeably. Let A be a $(0,1,2)$- matrix with m columns, let c be a site and let x be an element of $\{0,1,2\}$. Then $c(A,x)$ is defined as the set of rows of A containing the value x at site c. Let c and c' be sites and let x and y be elements of $\{0,1\}$. The pair c, c' *induces* (x,y) (in A) if $((c(A,x) \cap c'(A,y)) \cup (c(A,x) \cap c'(A,2)) \cup (c(A,2) \cap c'(A,y)) \neq \phi$. Let $IND(A,c,c')$ be the set of pairs (x,y) such that (c,c') induces (x,y) in A.
Note that, if B is a legal extension of A then, for every pair (c,c') of sites, IND(A,c,c')$\subseteq IND(B,c,c')$.
The following lemma has been proven independently by several authors:

LEMMA 2.1. *A $(0,1)$- matrix B is realizable and corresponds to a perfect phylogeny tree with root $(x_1, x_2, \cdots, x_m)$ if and only if, for every pair of sites (c_i, c_j), (c_i, c_j) induces (x_i, x_j) and $|IND(B, c_i, c_j)| \leq 3$.*

By Lemma 2.1, in the PPH problem, we have to assign $\{0,1\}$ values to the 2-entries in A so that the condition of the lemma is fulfilled for some z. Using our freedom to decide which of the nucleotides at a polymorphic site shall be designated 0 and which shall be designated 1, we may assume without loss of generality that the root of the tree is the all zeros vector for the following reason. By complementing certain columns (i.e., replacing 0 by 1 and 1 by 0 throughout the column) one can ensure that in each site c, either every row contains a 2 or a 0 appears in the first row not containing a 2. In this case, it is easy to see that unless there are two identical columns of 2's, every two sites induce $(0,0)$. It follows that, in any legal extension B for A, every two sites induce $(0,0)$. Thus, if B satisfies the condition of the lemma for some z, it also satisfies the condition for the all-zero vector.
In view of this observation we may restate the PPH problem as follows: Given a $(0,1,2)$-matrix A in which every pair of columns induces $(0,0)$, find a realization B of A such that, for every pair (c,c'), $|IND(B,c,c')| \leq 3$, or determine that no such realization exists. An immediate necessary condition for a realization to exist is that, for every pair (c,c'), $|IND(A,c,c')| \leq 3$.
Let the matrix A be an input to the PPH problem, and let B be a legal extension of A. Let c and c' be two columns such that $c(A,2) \cap c'(A,2) \neq \phi$. Let us say that B *resolves* the pair of columns (c,c') *unequally* if $\{(0,1),(1,0)\} \subseteq IND(B,c,c')$ and *equally* if $(1,1) \in IND(B,c,c')$.
Then B must resolve the pair (c,c') either equally or unequally, and cannot resolve the pair both equally and unequally. Solving the PPH problem is equivalent to deciding in a consistent way which pairs of columns to resolve equally and which to resolve unequally. These decisions essentially determine the matrix B.

3. THE BUILD-TREE ALGORITHM

In this section we present an algorithm for the PPH problem which runs in $O(nm^2)$ time.
Let A be a $(0,1,2)$-matrix with m columns such that, for every pair (c,c') where c and c' are not identical and $c(A,1) \cap c'(A,1) \neq \phi$, $(0,0) \in IND(A,c,c')$ and $|IND(A,c,c')| \leq 3$. We define the following relations:

Strong Domination $c \succ c'$ if IND(A,c,c') $= \{(0,0),(1,0),(1,1)\}$;

Siblings $c \sim c'$ if IND(A,c,c') $= \{(0,0),(0,1),(1,0)\}$;

Weak Domination $c \succeq c'$ if IND(A,c,c') $= \{(0,0),(1,0)\}$ or IND(A,c,c') $= \{(0,0)\}$.

Then for each pair (c,c'), exactly one of the following holds: $c \succ c'$, $c' \succ c$, $c \sim c'$, $c \succeq c'$, $c' \succeq c$.
Note that if $c \succ c'$, then c and c' must be equally resolved and, in a perfect phylogeny tree realizing A, c must be an ancestor of c'. If $c \sim c'$, then c and c' must be unequally resolved, and they must be siblings in the perfect phylogeny tree. The only ambiguous case is when $c \succeq c'$ or $c' \succeq c$.

3.1 The Main Algorithm

A $(0,1,2)$-matrix A with m columns is the input to the algorithm. The algorithm either determines that A is not realizable or:

1. Resolves certain rows of A into haplotypes, creating a new matrix M which is realizable if and only if A is;

2. Deletes one or more columns from M to create a matrix A' which is realizable if and only if A is, and such that any legal extension of A' yields a legal extension of A.

3. Recursively applies the algorithm to A'.

The algorithm begins by removing any identical columns c, c' in A, where $c(A,1) \cap c'(A,1) \neq \phi$. It then computes the relations between the sites and verifies that, for each pair (c,c'), $(0,0) \in IND(A,c,c')$ and $|(IND(A,c,c')| \leq 3$.
The algorithm then chooses a *pivot column* $\hat{c}$ as follows. Since strong domination and weak domination both induce a linear order on the columns, $\hat{c}$ is a maximal element with respect to both orders. Clearly, $\hat{c}$ can be found in $O(m)$ time.
Let $D_{\hat{c}} = \{c | \hat{c} \succ c\}$, $S_{\hat{c}} = \{c | \hat{c} \sim c\}$, and $W_{\hat{c}} = \{c | \hat{c} \succeq c\}$.It is easily verified that every column except $\hat{c}$ lies in exactly one of these three sets, and that no row in $\hat{c}(A,2)$ contains a 1.
The algorithm now proceeds to resolve the rows in $\hat{c}(A,2)$ Every column in $D_{\hat{c}}$ must be resolved equally with $\hat{c}$. Every column in $S_{\hat{c}}$ must be resolved unequally with $\hat{c}$. It remains to decide which columns in $W_{\hat{c}}$ are to be resolved equally with $\hat{c}$ and which are to be resolved unequally with $\hat{c}$
To make this determination the algorithm constructs a graph $G_{\hat{c}}$ whose vertex set is the set of sites. There is an edge labeled 0 between $\hat{c}$ and each site in $D_{\hat{c}}$. There is an edge labeled 1 between $\hat{c}$ and each site in $S_{\hat{c}}$. If c and d are sites different from $\hat{c}$ then there is an edge between c and d labeled 0 if $\hat{c}(A,2) \cap c(A,2) \cap d(A,2) \neq \phi$ and $(1,1) \in IND(A,c,d)$. There is an edge between c and d labeled 1 if $\hat{c}(A,2) \cap c(A,2) \cap d(A,2) \neq \phi$ and $c \sim d$. There are no other edges except as specified above.

Lemma 3.1. *If two vertices in $G_{\hat{c}}$ are joined by an edge labeled* 0 *then they must be resolved equally. If two vertices in $G_{\hat{c}}$ are joined by an edge labeled* 1 *then they must be resolved unequally.*

A path or cycle in $G_{\hat{c}}$ is *odd-weight* if it contains an odd number of edges labeled 1 and *even-weight* otherwise. It follows from the lemma that two sites joined by an odd-weight path must be resolved unequally, two sites joined by an even-weight path must be resolved equally, and hence that A is unrealizable if $G_{\hat{c}}$ contains an odd-weight cycle. Thus the algorithm terminates with failure if an odd-weight cycle exists.

If a site is joined to $\hat{c}$ by an even-weight path then it must be resolved equally with $\hat{c}$ and if it joined to $\hat{c}$ by an odd-weight path then it must be resolved unequally with $\hat{c}$.

It remains to determine how sites shall be resolved that lie in a different component of $G_{\hat{c}}$. The vertex set of each such component partitions uniquely into two parts, such that any two vertices in the same part must be resolved equally and any two vertices in different parts must be resolved unequally. For each such component the algorithm arbitrarily resolves the vertices in one of these parts equally with $\hat{c}$ and the vertices in the other part unequally with $\hat{c}$.

These choices uniquely determine the pair of haplotypes into which each row in $\hat{c}(A,2)$ shall be resolved. The algorithm replaces each such row by the vectors of these two haplotypes, creating a matrix M. It then deletes the columns in $\{\hat{c}\} \cup W_{\hat{c}}$ from M. Call the resulting matrix A'.

Theorem 3.2. *A' is realizable if and only if A is realizable.*

Proof. If $G_{\hat{c}}$ contains no odd-weight cycle then the algorithm resolves the rows in $\hat{c}(A,2)$ without creating a conflict. For any site $c \in W_{\hat{c}}$, $c(A,2) \subseteq \hat{c}(A,2)$. Therefore, once the rows in $\hat{c}(A,2)$, have been resolved M does not contain any 2's in the set of columns $\{\hat{c}\} \cup W_{\hat{c}}$. Therefore, resolution of the rows in M cannot create conflicts in any pair of columns containing at least one column from $\{\hat{c}\} \cup W_{\hat{c}}$. Also, for each row of M, the values in the columns of $\{\hat{c}\} \cup W_{\hat{c}}$ are determined independently of how the remaining columns are resolved. Therefore there is no loss of generality in dropping those columns from M, so M is realizable if and only A' is realizable. Finally the resolution of the columns of $D_{\hat{c}} \cup S_{\hat{c}}$ was uniquely determined by the requirement that columns in $D_{\hat{c}}$ be resolved equally with $\hat{c}$ and columns in $S_{\hat{c}}$ be resolved unequally with $\hat{c}$. Therefore A' is realizable if and only if A is realizable. $\square$

In order to implement the algorithm efficiently, we first precompute the relations between all pairs of sites, which takes $O(nm^2)$ time. After the preprocessing, each time we call the algorithm, we have to compute the edges of the graph and update the relations, which takes time $O(m^2)$ for each row that gets resolved. Thus, over the course of the algorithm, this work is $O(nm^2)$. Furthermore, at each iteration, we have to find a maximal site $\hat{c}$, which takes $O(m)$ time. We have to construct the graph $G_{\hat{c}}$, to determine how to resolve each pair of sites, and resolve the rows in $\hat{c}(A,2)$. This can be done in time $O(m^2|\hat{c}(A,2) \cap T|)$, where T is the set of rows that are not determined yet. Since each time it is done, the set $\hat{c}(A,2) \cap T$ is disjoint from previous such sets, the total running time is $O(nm^2)$. We note that the asymptotic running time in Bafna et. al., 2002 [2] is also $O(nm^2)$, and the asymptotic running time of Gusfield 2002 [13] is $O(nm\alpha(n,m))$, where α is the inverse Ackerman function.

In order to get all the possible legal extensions, we can change the algorithm as follows. Note that the only case where the assignment is not uniquely determined is when we have a connected component C of $G_{\hat{c}}$ that is contained in $W_{\hat{c}}$. Each such component partitions uniquely into two parts, such that the vertices in one part are resolved equally with $\hat{c}$, and the vertices in the other part are resolved unequally with $\hat{c}$. We introduce a boolean variable x_C which is an indicator of which part is labeled equally with $\hat{c}$. In the haplotypes resulting from the resolution of the rows in $\hat{c}(A,2)$ the entries in columns of $W_{\hat{c}}$ are recorded parametrically in terms of this boolean variable. These entries have no effect on the rest of the computation, since these columns are immediately deleted. It is easy to see that there is a $1-1$ correspondence between legal extensions and assignments of values to the boolean variables arising in this way. This gives a non-trivial upper bound of 2^{m-1} on the number of possible legal extensions. This bound is tight since, when all the entries of A equal 2, we have exactly 2^{m-1} solutions. We note that the algorithms presented in Gusfield 2002 [13], and in Bafna et. al. [2], also find the set of possible solution.

We show how to extend the algorithm to take into account known relations between some of the individuals in Appendix A.

4. IMPERFECT PHYLOGENY

In practice, the perfect phylogeny model is not effective in resolving haplotypes because the actual haplotypes do not fit the prefect phylogeny model. By Lemma 2.1, given a matrix B containing the haplotypes for a population of individuals, any pair of columns c and c' conflict with the perfect phylogeny model if $|IND(B,c,c')| = 4$. For example, consider sites 1 and 5 in Table 1. Note that the conflict arises when considering the sixth haplotype. If we consider the uncommon haplotypes in Table 1, there are many conflicts with the perfect phylogeny model. The sixth haplotype causes conflicts with site 5 and sites 1, 2, and 4. The seventh haplotype causes conflicts with site 5 and 2. Note that these conflicts only occur if we consider the uncommon haplotypes. Figure 2A shows the histogram of percentage of conflicts for the 11 blocks from Daly et al., 2001 [5]. The percentage of conflicts is defined by counting the number of pairs of columns that have conflicts and dividing by $\binom{n}{2}$.

In general, we can use the frequencies of the haplotypes to help cope with the presence of conflicts with the model. Consider sites 1 and 5 in Table 1. 24 haplotypes have a mutation at site 5 and no mutation at site 1. 10 haplotypes have a mutation at site 1 and no mutation at site 5. Only a single haplotype has a mutation at both site 1 and site 5. In this case, there is much more evidence to support that sites 1 and 5 should be resolved unequally rather than equally.

We extend our notion of "induces" to take into account an error threshold to handle the uncommon haplotypes. The pair c, c' *induces* (x,y) with error threshold k (in A) if $|((c(A,x) \cap c'(A,y)) \cup (c(A,x) \cap c'(A,2)) \cup (c(A,2) \cap c'(A,y))| > k$. Let $IND_k(A,c,c')$ be the set of pairs (x,y) such that (c,c') induces (x,y) with error threshold k in A. If we consider an error threshold of 1 for Table 1 then there are no conflicts to the perfect phylogeny model.

Figures 2B-C show the percentage of conflicts for the 11 blocks in the data from Daly et al., 2001 [5] with a 5% and 10% error threshold. Clearly, as the error threshold increases, the number of conflicts significantly decreases. This is due to the fact that the infrequent haplotypes cause the majority of the conflicts with the perfect phylogeny model.

4.1 Handling Noise in the Data

In practice, the biological data does not exactly fit the perfect phylogeny model. We therefore pose the problem of removing the minimal number of individuals from our data set so that the remaining data fits the perfect phylogeny model. However, this is a hard problem. In the final version we provide hardness result for this problem.

We apply three heuristics modifying the algorithm to relax the constraints of the perfect phylogeny model. The first heuristic handles the conflicts caused by the uncommon haplotypes by using the error threshold which is set to a fraction of the data. Even with the error threshold, in some cases in a given matrix A, a pair of sites c and c' may have a conflict such that $|IND_k(A, c, c')| = 4$. The second heuristic redefines the notion of "induces" in these cases by removing one of the pairs $\{(0, 1), (1, 0), (1, 1)\}$ from $IND_k(A, c, c')$ and removing the conflict. The pair that is removed is chosen based on having the least number of individuals in the matrix A support the pair. The third heuristic assigns a notion of strength to each edge in the graph $G_{\hat{c}}$. The strength of an edge between c and d different from $\hat{c}$ is $|\hat{c}(A, 2) \cap c(A, 2) \cap d(A, 2)|$. If $G_{\hat{c}}$ contains an odd-weight cycle, we repeatedly remove the weakest edge until no cycle exists.

5. MAXIMUM LIKELIHOOD MODEL FOR LOCAL HAPLOTYPE PHASE RESOLUTION

We choose the "best" solution from the set of candidate solutions that roughly fit the perfect phylogeny model using a maximum likelihood model. The maximum likelihood model estimates the likelihood of observing the population of genotypes given the predicted haplotype frequencies. The likelihood model assumes independence between the haplotypes of an individual.

Given a population of n individuals, we denote the two haplotypes of the ith individual as i_1 and i_2. We use the notation $f(i_1)$ to denote the frequency of the haplotype i_1 in the population. The likelihood of a haplotype i_1 is $\frac{f(i_1)}{2n}$. The likelihood for each genotype of an individual is simply the product of the likelihoods of their two haplotypes $\frac{f(i_1)f(i_2)}{(2n)^2}$. The likelihood of a candidate solution for a population of genotypes is

$$L = \prod_{i=1}^{n} \frac{f(i_1)f(i_2)}{(2n)^2} \tag{1}$$

This model is consistent with previous maximum likelihood models for choosing haplotypes from genotypes [7, 14, 17, 8, 26]. The main caveat with these previous approaches is that they do not restrict the possible solutions to the ones that roughly fit the perfect phylogeny model. This results in far to many possible haplotype resolutions that need to be evaluated using the maximum likelihood model, and thus the previous algorithms are inefficient and not practical.

5.1 Resolving Missing Data

Missing data is resolved after the algorithm resolves heterozygous genotypes. When determining the relations between sites, individuals with missing genotype data for one of those sites are ignored. Once the maximum likelihood model chooses the best haplotype resolutions, there are typically several common haplotypes which account for the majority of the population. At this point missing genotypes are resolved by choosing the most likely SNP based on the maximum likelihood model. Effectively, we resolve the missing data by choosing the SNP to match the common haplotypes.

6. COMPUTING BLOCK PARTITIONS FROM GENOTYPES

Since there are only a few haplotypes in each block, we do not need to check every SNP in the block to determine which of the common haplotypes an individual has. For each block, we can define a set of representative SNPs that are sufficient to determine an individual's haplotypes. In Table 1, the second, third and fifth SNPs are sufficient to determine the haplotype. For example, if we observe T, A, and T in these SNPs, we can infer that the individual has the third haplotype (assuming the individual has one of the common haplotypes). On the other hand, if we observe T, G, and C, we can infer the individual has the second haplotype. For this block, there are other possibilities for a set of representative SNPs such as the first, second and fifth SNPs or the second, fourth and fifth haplotypes. However, for this block, the minimum number of representative SNPs is three. That is, no two SNPs can distinguish the four common haplotypes.

A criterion for determining a good block partition is minimizing the sum of the number of representative SNPs over all blocks. This criterion has been used to partition SNPs into blocks on a larger scale [22, 27]. Our method predicts block partitions directly from the genotype data. We first define a set of candidate blocks. Given a maximum block length, we slide a window across the data for each block length to define our candidate blocks. For each candidate block, we apply the local haplotype prediction algorithm to predict the haplotypes. Our algorithm accurately predicts haplotypes only if there is limited diversity within a block. To ensure accuracy of our predictions, we discard all candidate blocks that have more than five common haplotypes. For each remaining candidate block, we determine the number of representative SNPs. This is done by enumerating over all subsets of the SNPs in the block and checking to see if they distinguish between the common haplotypes.

To compute the block boundaries for the haplotypes, we use a straightforward dynamic programming technique similar to the one presented in [27]. The main difference is that in our setting, there is no missing data since it is resolved by the local prediction algorithm. Note that the block partition in Daly et al., 2001 [5] does not assign several SNPs to blocks. We can easily modify the dynamic programming algorithm to optimize a block partition where several SNPs are allowed to be left out.

7. TILING LOCAL BLOCKS TO OBTAIN GLOBAL HAPLOTYPE STRUCTURE

We use a "tiling" technique to extend the haplotype predictions across block boundaries. We make predictions for a set of "tiling" blocks. These blocks span our original block boundaries. For each haplotype block boundary, we make a predictions for a tiling block of length 6 which spans the boundary and 3 SNPs on either side. At a block boundary, we have the predictions for both haplotypes on each side of the boundary as well as the predictions for the haplotypes

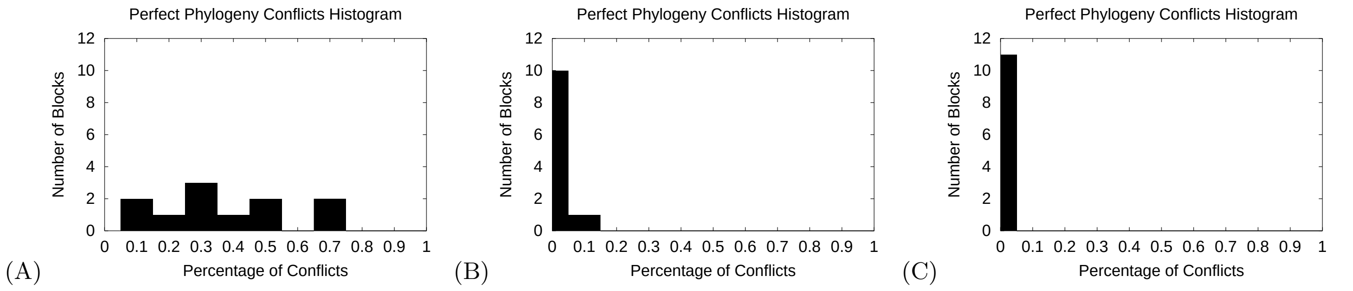

Figure 2: Histograms of percentage of conflicts under different error thresholds for the blocks defined in Daly et al., 2001 [5]. Thresholds are (A) 0% (B) 5% and (C) 10%.

of the tiling block that spans this boundary. There are four possible ways to arrange these six blocks on the two chromosomes. We choose the arrangement that has the least number of inconsistencies. In some cases it is impossible to determine how to connect the blocks because two possibilities have the minimum number of inconsistencies. We refer to these cases as *unresolvable*. Consider the case where one of the blocks contains two copies of the same haplotype. Since the haplotypes are the same, it is impossible to determine to how to connect the neighboring haplotypes.

Consider the following example where an individual has the haplotypes $TACGC$ and $TACGT$ for predicted block 2 in Table 2 and $CGGAGACGA$ and $GACTGGTCG$ for block 3. The prediction are $CGCGAC$ and $CGTCGG$ for the tiling block. Using these predictions, we deduce that the complete haplotypes are $TACCGCGACTGGTCG$ and $TACGTCG$-$GAGACGA$. Note that the actual predictions of the tiling block may not be as accurate as the block predictions. However, as the results in section 8.3 show, they are accurate enough to make a decision between two choices on how to join together the haplotypes.

8. RESULTS

Our first set of experiments assumes that we are given the block partition for data. Our second set of experiments assumes that we have no prior information about the block partition and are only given the genotypes. We apply our algorithm to determine the block partition. We also use a tiling technique to extend the haplotype predictions across block boundaries. We evaluate our predictions by comparing them to the correct haplotypes inferred from the trios.

The data set over which we perform our experiments is a 500 kilobase region of chromosome 5p31 containing 103 SNPs from the studies of Daly et al., 2001 [5] and Rioux et al., 2001 [24]. In this study, genotypes for the 103 SNPS are collected from 129 mother, father, child trios from a European-derived population in an attempt to identify a genetic risk factor for Crohn's disease. A significant portion of the genotype data (10.03%) is missing with an average of 10 SNPs per individual's genotype missing. The 103 SNPs were split into 11 blocks containing from 5 to 31 SNPs and ranging from 3 to 92 kilobases. For each of these blocks, four haplotypes correspond to 90% of the individual chromosomes. Since this set consists of trios, we can infer each individual's haplotypes. This data is publicly available at `http://www-genome.wi.mit.edu/humgen/IBD5/`.

To evaluate our predictions of haplotypes, we make predictions over the genotype data of the individuals and then compare our predictions to the correct haplotypes inferred from the trios.

8.1 Predicting Haplotypes within a Block

In the first set of experiments, for each of the 11 blocks as defined in Daly et al., 2001 [5] we predicted the common haplotypes from the genotypes of the children in the trios as well as each child's haplotype. In all cases but the fourth block, the predictions for the common haplotypes are consistent with the published predictions. A significant portion of the data is missing, 10.03% of the total genotype data. This missing data comes from various sources of experimental error. We resolve the missing data. Over the 129 individuals and 103 SNPs, our error rate is only 0.53% in terms of bases, significantly lower the the amount of missing genotype data. If we ignore the missing data, our error rate in terms of bases is only 0.30%. Over individuals that contain the common haplotypes, our predictions are perfect. The errors only occur for individuals who have an uncommon haplotype. The program takes only a few seconds to make each of these haplotype predictions.

8.2 Predicting Blocks from Genotypes

Typically, we must determine the block partition directly from the genotype data. We first make haplotype predictions for all possible blocks of up to length 30 using the local haplotype prediction algorithm and discard any blocks with more than five common haplotypes. This leaves 1140 potential blocks. We note that the number 30 could be easily changed. As long as the block size is in the order of magnitude of 100, our algorithm proves to be practically efficient. In the data given by Daly et. al, 2001 [5] and in the data given by Patil et. al., 2001 [22](which includes the entire chromosome 21, there were no blocks of much larger length than 100 (see Zhang et. al.,2002 [27]).

Using dynamic programming, we choose the best block partition for the data from Daly et al. 2001, [5] where the objective is to minimize the number of representative SNPs over the entire block partition. Table 2 shows the predicted block partition which contains 27 representative SNPs. For each block in the partition, Table 2 gives the number of representative SNPs in the block, the common haplotypes, as well as the error rate after resolving the missing data. Over the blocks chosen by the algorithm, the error rate for the 103 SNPs is 0.73% and only 0.47% if we ignore the missing data. The block partition varies from the partition described in Daly et al., 2001 [5] since the criteria for defining block partitions vary.

SNPs	Predicted Common Haplotypes	Freq.	Error Rate
1-10	GGACAACCGT	199	0.0016
	AATTCGTGGC	34	
11-15	TACGC	134	0.0108
	TACGT	85	
	CCAAC	34	
16-24	CGGAGACGA	139	0.0078
	GACTGGTCG	52	
	CGCAGACGA	34	
25-36	CGCGCCCGGATC	141	0.0019
	CTGCCCCGGCTC	35	
	CTGCTATAACCG	34	
	TTGCCCCAACCC	23	
37-46	CAGCCCGATC	139	0.0101
	CGCTCTGACT	36	
	CACCATACTC	29	
	CACCCTGACT	18	
	AACCCTGACC	14	
47-76	CCTGCTTACGGTGCAGTGGCACGTATTGCA	137	0.0044
	CCCATCCATCATGGTCGAATGCGTACATTA	58	
	CCCGCTTACGGTGCAGTGGCACGTATATCA	21	
	ATCACTCCCCAGACTGTGATGTTAGTATCT	12	
	CCTGCTTACGGTGCAGTGGCACGTATTTCA	9	
77-79	CCG	151	0.0232
	GTG	51	
	CTG	40	
80-82	TTT	205	0.0000
	ATT	29	
83-91	AGCACAACA	144	0.0121
	GACGCGGTG	48	
	GAGGCGGTG	18	
	GGCGTGACG	13	
	AGCACGGTG	13	
92-98	GTTCTGA	142	0.0078
	TGTGTAA	49	
	TGTGCGG	33	
	TGCGTAA	15	
99-103	CGGCG	112	0.0031
	TATAG	105	
	TATCA	35	

Table 2: Predicted block partition over data from Daly et al. 2001, [5]. The second column gives the number of representative SNPs. The third column shows the predictions for the common haplotypes and the fourth gives their frequencies. The fifth column gives the error rate after resolving missing data. The error rate is the total number of errors in the predictions for the divided by the total number of bases in the block. The error rate includes predictions for the uncommon haplotypes. The overall error rate over the 103 SNPs resolving missing data is 0.73%.

8.3 Tiling Block Predictions

Over the predicted blocks, for each individual, we can accurately predict which haplotypes the individual contains for each block. A more difficult problem is to recover the complete haplotypes of the individual. Given the haplotype block predictions, this problem reduces to determining which of the individuals' haplotypes are on one chromosome and which are on the other. At each block boundary, there are 2 possibilities for how the haplotype blocks are arranged on the chromosome. Using the predictions of Table 2, since there are 11 blocks and each of the two predicted haplotypes can be on either chromosome, there are a total of 2^{10} possible complete haplotypes.

For each individual in the data, we must make 10 choices on how to connect their blocks. Over the data set of 129 individuals, 784 of these choices are unresolvable. Over the remaining 506 choices, we correctly predict 473 of them for an error rate of 6.5%.

8.4 Scaling to Chromosome Size Samples

The actual problem of reconstructing the block partition over a chromosome will involve the partitioning of massive number of SNPs. In order to measure how our algorithm scales to large data sets of this size, we applied our algorithms to a larger sample from Patil et al., 2001 [22] where 24,047 SNPs over a 32.4 megabase region of chromosome 21 are split into 4135 blocks. The sample contained 20 haplotypes. We generated a simulation genotype dataset as follows. We generated 200 individuals where at each block position one of the 20 haplotypes from the sample was chosen at random. We then measured the throughput of the algorithm over this sample. On a 750 MHz processor, the algorithm was able to partition the SNPs into blocks and predict the haplotypes at a rate of approximately 1,000 SNPs per hour making it practical for handling data sets of this size. The algorithm predicted correctly the block partition and the haplotypes but this is expected due to the fact that only 20 distinct haplotypes at each block position.

8.5 Comparison with PHASE

We applied the widely used program PHASE [26] to reconstruct the haplotypes in order to provide a comparison with our method. PHASE is able to make local predictions for each of the 11 blocks from Daly et al., 2001 [5], however, these predictions took hours for each block. In practice, since we do not know the block partition, we must make predictions for each of more than 2,500 candidate blocks in order to determine the optimal block partition. This makes PHASE impractical for predicting the block partition from the genotypes.

9. DISCUSSION

Recent studies in haplotype structure have shown that haplotypes are structured into blocks with limited diversity. A recent paper by Gusfield 2002 [13] suggested the use of perfect phylogeny to reconstruct the haplotypes. In this paper, we have presented a practical algorithm for reconstructing haplotype information from genotype data that leverages both the block structure and the perfect phylogeny model.

We have demonstrated our method over actual haplotype data collected from 129 trios and verified the accuracy of our predictions to the correct haplotypes. Our method is highly accurate and efficient. The predictions differ from the actual haplotypes by less than 1% even after resolving approximately 10% of the missing genotype data. We also present a method for determining the block partition from genotype data and a method for extending haplotype predictions beyond single blocks.

The program for predicting haplotype structure is publicly available via a webserver at
`http://www.cs.columbia.edu/compbio/haplotype/`

10. REFERENCES

[1] M.F. Plass B. Aspval and R.E. Tarjan. A linear time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letter*, 8:121–123, 1979.

[2] V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph. Haplotyping as perfect phylogeny: A direct approach. *Technical Report UCDavis CSE-2002-21*, Jul 17 2002.

[3] AG Clark. Inference of haplotypes from pcr-amplified samples of diploid populations. *Journal of Molecular Biology and Evolution*, 7(2):111–22, Mar 1990.

[4] S. A. Cook. The complexity of theorem-proving procedures. *STOC*, pages 151–158, 1971.

[5] 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.

[6] Eleazar Eskin, Eran Halperin, and Richard M. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. *Technical Report. UC Berkeley Computer Science*, 2002.

[7] L Excoffier and M Slatkin. Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Molecular Biology and Evolution*, 12(5):921–7, Sept 1995.

[8] D Fallin and NJ Schork. Accuracy of haplotype frequency estimation for biallelic loci, via the expectation-maximization algorithm for unphased diploid genotype data. *American Journal of Human Genetics*, 67(4):947–59, Oct 2000.

[9] DB Goldstein and ME Weale. Population genomics: linkage disequilibrium holds the key. *Current Biology*, 11:R576–R579, 2001.

[10] The International SNP Map Working Group. A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. *Nature*, 409(6822):928–33, 2001.

[11] D Gusfield. A practical algorithm for optimal inference of haplotypes from diploid populations. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, 2000.

[12] D Gusfield. Inference of haplotypes from samples of diploid populations: complexity and algorithms. *Journal of Computational Biology*, 8(3):305–23, 2001.

[13] Dan Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions (extended abstract). In *Proceedings of the 6th International Conference on Computational Molecular Biology (RECOMB 2002)*, 2002.

[14] ME Hawley and KK Kidd. Haplo: a program using the em algorithm to estimate the frequencies of multi-site haplotypes. *Journal of Heredity*, 86(5):409–11, Sep-Oct 1995.

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

[16] G. Lancia, V. Bafna, S. Istrail, R. Lippert, and R. Schwartz. Snps problems, algorithms and complexity, european symposium on algorithms. In Springer-Verlag, editor, *Proceedings of the European Symposium on Algorithms (ESA-2001), Lecture Notes in Computer Science*, volume 2161, pages 182–193, 2001.

[17] JC Long, RC Williams, and M Urbanek. An e-m algorithm and testing strategy for multiple-locus haplotypes. *American Journal of Human Genetics*, 56(3):799–810, Mar 1995.

[18] V.V. Vazirani N. Garg and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM J. Comp*, 25:235–251, 1996.

[19] NIH. Large-scale genotyping for the haplotype map of the human genome. RFA: HG-02-005.

[20] C.H. Papadimitriou. On selecting a satisfying truth assignment. *FOCS*, pages 163–169, 1991.

[21] C.H. Papadimitriou and M.Yannakakis. Optimization, approximation and complexity classes. *JCSS*, 43:425–440, 1991.

[22] N Patil, AJ Berno, DA Hinds, WA Barrett, JM Doshi, CR Hacker, CR Kautzer, DH Lee, C Marjoribanks, DP McDonough, BT Nguyen, MC Norris, JB Sheehan, N Shen, D Stern, RP Stokowski, DJ Thomas, MO Trulson, KR Vyas, KA Frazer, SP Fodor, and DR Cox. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, 294(5547):1719–23, Nov 23 2001.

[23] DE Reich, M Cargill, S Bolk, J Ireland, PC Sabeti, DJ Richter, T Lavery, R Kouyoumjian, SF Farhadian, R Ward, and ES Lander. Linkage disequilibrium in the human genome. *Nature*, 411(6834):199–204, May 10 2001.

[24] JD Rioux, MJ Daly, MS Silverberg, K Lindblad, H Steinhart, Z Cohen, T Delmonte, K Kocher, K Miller, S Guschwan, EJ Kulbokas, S O'Leary, E Winchester, K Dewar, T Green, V Stone, C Chow, A Cohen, D Langelier, G Lapointe, Gaudet D, J Faith, N Branco, SB Bull, RS McLeod, AM Griffiths, A Bitton, GR Greenberg, ES Lander, KA Siminovitch, and TJ Hudson. Genetic variation in the 5q31 cytokine gene cluster confers susceptibility to crohn disease. *Nature Genetics*, 29(2):223–8, Oct 2001.

[25] A. Itai S. Even and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SICOMP*, 5:691–703, 1976.

[26] 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.

[27] K Zhang, M Deng, T Chen, MS Waterman, and F Sun. A dynamic programming algorithm for haplotype block partitioning. *Proceedings of the Nationall Acadamy of Science*, 99(11):7335–9, May 28 2002.

APPENDIX

Appendix

A. ADDING FAMILIES TO THE DATA

In many cases, the experimental studies are done on related individuals, such as a set of trios of mother, father and a child. In this case, in addition to the perfect phylogeny model, we have the additional constraint that each of the parents transmits exactly one of its haplotypes to the child. In this section we show that this extension to the PPH problem can be solved in polynomial time. Formally, we solve the following problem:

A.0.0.1 The Trios PPH problem (TPPH)..

The input is a $\{0, 1, 2\}$ matrix $A = (a_{ij})$ of size $n \times m$ and a set of triplets $T \subseteq [n]^3$. Every triplet $(r_1, r_2, r_3) \in T$ represents a mother father and child trio. We need to determine if there is a legal extension $B = (b_{ij})$ to A such that for every triplet $(r_1, r_2, r_3) \in T$, one of the rows $B(2r_3 - 1, :), B(2r_3, :)$ is a duplicate of one of the rows $B(2r_1 - 1, :), B(2r_1, :)$, and the other is a duplicate of one of the rows $B(2r_2 - 1, :), B(2r_2, :)$. This correspond to the fact that one of the haplotypes is transmitted from the mother and the other from the father.

A.1 Solving TPPH

Recall that the output of algorithm *Build-Tree* is a matrix B which corresponds to all possible solutions to the perfect phylogeny model. For each $i \in [2n], j \in [m]$, either $b_{ij} \in \{0, 1\}$, or $b_{ij} \in \{x_1, \ldots, x_k, \bar{x_1}, \ldots, \bar{x_k}\}$, where x_i is a boolean variable for each $i \in [k]$. We have to set the values of the variables, so that the solution will be consistent with the

trios.
For ease of notation, for every triplet $t = (r_1, r_2, r_3) \in T$, and every column i, we say that the (t, i) configuration is $(z_1, z_2, z_3, z_4, z_5, z_6)$ if $z_1 = b_{2r_1-1,i}, z_2 = b_{2r_1,i}, z_3 = b_{2r_2-1,i}, z_4 = b_{2r_2,i}, z_5 = b_{2r_3-1,i}, z_6 = b_{2r_3,i}$, that is, the values z_1, z_2, z_3, z_4, z_5 and z_6 represent the values of the mother, father and child's haplotypes. We first need the following lemma.

Lemma A.1. *For every triplet $t \in T$ and every column $c \in [m]$, the following (t, c) configurations never appear in B:*

1. *$(1, 1, *, *, x, \bar{x})$ for $x \in \{x_1, \ldots, x_k, \bar{x_1}, \ldots, \bar{x_k}\}$, where $*$ represent an arbitrary value. Any permutations of the values between the mother, father and child does not appear either.*
2. *$(x, \bar{x}, y, \bar{y}, *, *)$ where $x \in \{x_1, \ldots, x_k, \bar{x_1}, \ldots, \bar{x_k}\}$, and $x \neq y$. Any permutations of the values between the mother, father and child does not appear either.*

For each triplet $t = (r_1, r_2, r_3) \in T$ we introduce three additional boolean variables $tr_1(t), tr_2(t), p(t)$ which have the following values:

- $p(t) = 1$ if and only if the haplotype $B(2r_3 - 1, :) = B(2r_1, :)$ or $B(2r_3 - 1, :) = B(2r_1 - 1, :)$. In other words, $p(t) = 1$ if and only if the first haplotype of r_3 is transmitted from the mother.
- $tr_1(t) = 1$ if and only if $B(2r_1 - 1, :) = B(2r_3 - 1, :)$ or $B(2r_1 - 1, :) = B(2r_3, :)$, that is, if the first haplotype of the mother is transmitted to the child.
- $tr_2(t) = 1$ if and only if $B(2r_2 - 1, :) = B(2r_3 - 1, :)$ or $B(2r_2 - 1, :) = B(2r_3, :)$, that is, if the first haplotype of the child is transmitted to the child.

Whenever it is clear from the context, we will omit t, and just write p, tr_1, tr_2 instead of $p(t), tr_1(t), tr_2(t)$. It is easy to see that any $\{0, 1\}$ assignment to the variables $p(t), tr_1(t), tr_2(t)$ corresponds to one of the eight possible transmissions of the haplotypes from the parents to the child. We thus get that the TPPH problem is equivalent to assigning $\{0, 1\}$ values to the variables $x_1, \ldots, x_k$, and the variables $p(t), tr_1(t), tr_2(t)$ for $t \in T$. Every triplet $t \in T$, and every column $j \in [m]$, impose a constraint on the variables. For example, if the mother has 0 values in both haplotypes, the father has 1 in both haplotypes, and the child has 1 in the first haplotype, and 0 in the second haplotype, then clearly, $p(t) = 1$. We will now show that all these constraints can be represented as a 2-CNF formula, and thus, can be solved using any polynomial algorithm known for 2-SAT. In Table 3 we list all possible constraints which follow from the (t, c) configurations. It is easy to verify by Lemma A.1, that all the possible configurations are listed in the table, up to symmetry. Note that since for any two boolean variables x, y $x = y$ can be expressed as $(x \vee \bar{y}) \wedge (\bar{x} \vee y)$, and $x \neq y$ is $x = \bar{y}$, we get that all the constraints in Table 3 can be expressed as 2-CNF constraints.
We now describe an algorithm for solving the TPPH problem. We first use algorithm Build-Tree to get all possible solutions to the PPH problem. We now let C be the set of constraints induced by all the triplets. Since can express all the constraints in C by a 2-CNF formula, we can a legal assignment to the variables by using any of the algorithms known for 2-SAT [1, 4, 25, 20]. If there is no feasible solution, we report that the problem is infeasible.

1	(z, z, z, z, z, z)	No constraint
2	$(x, \bar{x}, 0, 0, 0, 0)$	$x \neq tr_1$
3	$(z, \bar{z}, z, z, z, z)$	$tr_1 = 1$
4	$(z, \bar{z}, z, \bar{z}, z, z)$	$tr_1 \wedge tr_2$
5	$(x, \bar{x}, x, \bar{x}, 0, 0)$	$(tr_1 = tr_2) \wedge (tr_1 \neq x)$
6	$(z, \bar{z}, \bar{z}, z, z, z)$	$tr_1 \wedge \bar{tr_2}$
7	$(z, z, \bar{z}, \bar{z}, z, \bar{z})$	$p = 1$
8	$(x, \bar{x}, 0, 0, x, \bar{x})$	$x = tr_1 = p$
9	$(z, \bar{z}, z, z, z, \bar{z})$	$\bar{p} \wedge \bar{tr_1}$
10	$(z, \bar{z}, z, \bar{z}, z, \bar{z})$	$(tr_1 \neq tr_2) \wedge (tr_1 = p)$
11	$(x, \bar{x}, x, \bar{x}, x, \bar{x})$	$(tr_1 \neq tr_2) \wedge (tr_1 = p)$
12	$(\bar{z}, z, z, \bar{z}, z, \bar{z})$	$(tr_1 = tr_3) \wedge (tr_1 \neq p)$

Table 3: The possible constraints up to symmetry. z represents any value in $\{0, 1\}$. x represents a literal, that is, $x \in \{x_1, \ldots, x_k, \bar{x_1}, \ldots, \bar{x_k}\}$.