

# Efficient Reconstruction of Haplotype Structure via Perfect Phylogeny

Eleazar Eskin\*

Eran Halperin†

Richard M. Karp‡

## Abstract

Each person's genome contains two copies of each chromosome, one inherited from the father and the other from the mother. A person's *genotype* specifies the pair of bases at each site, but does not specify which base occurs on which chromosome. The sequence of each chromosome separately is called a *haplotype*. The determination of the haplotypes within a population is essential for understanding genetic variation and the inheritance of complex diseases. The haplotype mapping project, a successor to the human genome project, seeks to determine the common haplotypes in the human population.

Since experimental determination of a person's genotype is less expensive than determining its component haplotypes, algorithms are required for computing haplotypes from genotypes. Two observations aid in this process: first, the human genome contains short blocks within which only a few different haplotypes occur; second, as suggested by Gusfield, it is reasonable to assume that the haplotypes observed within a block have evolved according to a *perfect phylogeny*, in which at most one mutation event has occurred at any site, and no recombination occurred at the given region.

We present a simple and efficient polynomial-time algorithm for inferring haplotypes from the genotypes of a set of individuals assuming a perfect phylogeny. Using a reduction to 2-SAT we extend this algorithm to handle constraints that apply when we have genotypes from both parents and child. We also present a hardness result for the problem of removing the minimum number of individuals from a population to ensure that the genotypes of the remaining individuals are consistent with a perfect phylogeny.

Our algorithms have been tested on real data and give biologically meaningful results. Our web-server (<http://www.cs.columbia.edu/compbio/hap/>) is publicly available for predicting haplotypes from genotype data and partitioning genotype data into blocks.

## 1 Introduction

Critical to the understanding of the genetic basis for complex diseases is the modeling of human genetic 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 been passed on through heredity. Although the two chromosomes of an individual can be separated and analyzed independently as in the study of [21], current technology suitable for large scale polymorphism screening obtains *genotype* information at each SNP. The genotype gives the bases at the SNP for both copies of the chromosome, but does not identify 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 genotype. Two of the cases are where either both chromosomes contain  $A$  or both chromosomes contain  $G$ . We refer to these cases as *homozygous* genotypes. The other two cases are where the first chromosome contains  $A$  and the second contains  $G$  and vice versa. We refer to these cases as *heterozygous* genotypes. Thus, the genotype consists of the mutual information on the two chromosomes. The sequence of each chromosome separately is called the haplotype information. Consider a case where, at four successive SNPs, with possible values  $A$  or  $G$ , an

---

\*Computer Science Department, Columbia University. E-mail: [eeskin@cs.columbia.edu](mailto:eeskin@cs.columbia.edu).

†CS Division, Soda Hall, University of California Berkeley, CA 94720-1776. E-mail: [eran@eecs.berkeley.edu](mailto:eran@eecs.berkeley.edu).

‡International Computer Science Institute, 1947 Center St., Berkeley, CA 94704. E-mail: [karp@cs.berkeley.edu](mailto:karp@cs.berkeley.edu).

individual has a genotype  $AHHG$ , where  $H$  represents a heterozygous site. In this case, the individual's haplotypes have two possibilities: either one chromosome contains  $AAAG$  and the other contains  $AGGG$  or one chromosome contains  $AAGG$  and the other contains  $AGAG$ .

One of the first goals set by the NIH right after the completion of the human genome project is the haplotype mapping project. The goal of the human genome project was to find the consensus genotype sequence of humans. In order to achieve more information on genetic disease, one has to know not only the genotype data, but also the haplotype data, and not only the consensus, but which are the common haplotypes. Recent studies [6, 21] have shown that SNPs that are physically close to each other on the chromosome are usually correlated, and that our chromosomes can be partitioned into blocks, so that in each block there is a strong correlation between all the SNPs contained in it. These studies show that for each block, the number of possible variations is usually very small (3 or 4), while the number of SNPs in the block could be as large as 30. The goal of the haplotype mapping project is to gather all the haplotype information, that is, for each block, to list all possible combinations of SNPs that appear in the population. After having all this information, one could perform a more accurate case study to associate genes (and maybe blocks) with diseases, and furthermore, one could sequence the human chromosomes much faster with high accuracy. This paper takes the haplotype mapping project one step forward by finding an efficient way to infer the haplotype data of a population by observing only the genotypes of the population.

Given a set of  $n$  genotypes, each of length  $m$ , we address the problem of inferring the haplotype structure. Clearly, in the absence of additional information, one cannot infer the haplotype structure, since there are many possible solutions. Gusfield [14] suggested to add the constraint that the resulting set of haplotypes should correspond to a phylogeny model known as the coalescent model, or perfect phylogeny. In this model we assume that the chromosomes of each parent are transmitted to the child with some possible mutations, but with no recombination. Furthermore, we assume that in each SNP site, there has only been one mutation throughout the history represented by the tree. We wish to find a directed phylogenetic tree that will correspond to these two assumptions, such that each of the resulting haplotypes will be found in one of the leaves of that tree. The problem was introduced by Gusfield [14], and is referred to as the Perfect Phylogeny Haplotype problem (PPH problem). We will formally state the problem in Section 2.1. Gusfield [14] introduces a polynomial time algorithm for the PPH problem. His algorithm uses as a black box an algorithm to recognize graphic matroids [23, 3]. The asymptotic running time of his algorithm is  $O(nm\alpha(n, m))$ , where  $n$  is the number of individuals,  $m$  is the number of SNPs, and  $\alpha$  is the inverse ackerman function which is a very slowly increasing function, so that for all practical applications it could be considered as a constant. Although his algorithm is theoretically very efficient, and the matroid theory behind it is mathematically very elegant, the algorithm is very complicated and not easy to implement. One of the open problems mentioned in Gusfield's paper is to find a simple and efficient algorithm to the PPH problem. In this paper we introduce an efficient and simple solution to the problem, using no heavy machinery. The asymptotic running time of our algorithm is  $O(nm^2)$ , which is not as efficient as Gusfield's algorithm, but on the other hand, the simplicity of our algorithm sheds a new insight on the problem and allows us to cope with some extensions of the model. Furthermore, note that in practice  $m$  is usually of the order of a constant (between 5 and 30, and normally not more than 10), and thus, in practice, the simplicity of the algorithm gives a great advantage in the running time. We implemented the algorithm, and observe that in practice it is extremely efficient. We note that Bafna et. al. [2] independently found a different algorithm with the same theoretical performance as our algorithm. We further show relations between the extensions of the problem and other combinatorial problems such as 2-satisfiability and minimum CNF deletion problem.

We begin by presenting an extremely simple and elegant polynomial-time algorithm for the problem. Although this algorithm may be effective in practice its time bound is unreasonably high. Therefore we go on to present our main algorithm, which runs in  $O(nm^2)$  time, and produces a simple linear size data structure which can be used to produce all possible solutions to the problem. Each possible solution can be

implicitly enumerated in time  $O(m)$  (clearly, to output the solution one needs  $O(mn)$  time). We note that the algorithms of [2] and [14] also find a linear size data structure which implicitly stores all solutions. We furthermore extend our main algorithm to handle the additional constraint that some of the individuals are related, and therefore, a parent must transmit one of its haplotypes to a child, and each child has one haplotype transmitted from its father, and the other from its mother. We use the data structure returned by our main algorithm for the PPH problem as a starting point for the algorithm, and reduce the resulting problem to the 2-SAT problem which can be solved in polynomial time [1, 5, 7, 19].

Finally, we address the problem of finding a minimum set of individuals such that by removing them from our data set, we will be able to find at least one solution to the PPH problem. We show that finding an  $\alpha$ -approximation to this problem will imply an  $\alpha$ -approximation algorithm for the Min UnCut problem, where a graph  $G = (V, E)$  is given, and the goal is to remove the minimum number of edges in  $G$  such that the remaining graph is bipartite. This problem has a  $\log n$ -approximation algorithm by a reduction from the minimum 2-CNF deletion problem [10]. On the negative side it is only known that the problem is MAX-SNP hard, and therefore there is no PTAS for the problem [20] unless  $P=NP$ . We note that the same problem when the input does not contain heterozygous sites, i.e. given a  $\{0, 1\}$  matrix, removing the minimum number of rows, so that the resulting matrix fits the perfect phylogeny model, can be easily approximated by a factor of 3.

We evaluate our algorithm over the data collected in the study of a 500 kilobase region of chromosome 5p31 containing 103 SNPs from the study of [6]. In this study, genotypes are collected from 129 mother, father, child trios and the correct child haplotypes are inferred from these genotypes. In our experiments, we use our method to make predictions of the child's haplotypes from the child's genotypes and then check our predictions against the correct haplotypes inferred from the trios. Our results indicate that the algorithm is practical and efficient, and gives biologically meaningful results.

There are several previous approaches to determining the haplotype information from genotype data. These methods include the parsimony approach of Clark [4] and related approaches [12, 13, 17], maximum likelihood methods [8, 16, 18, 9] and statistical methods such as PHASE [22]. These approaches use heuristics, but in practice they do not scale to data that contains more than 30 sites, while our algorithm can cope with large data sets.

## 2 Preliminaries

We first formally describe the coalescent model (perfect phylogeny) 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  $n \times m$  which represents a set of haplotypes, we say that  $B$  fits the coalescent 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  $T(B)$ .

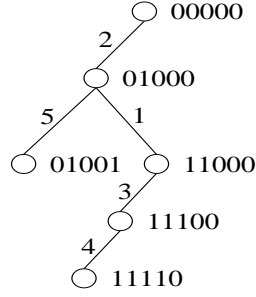


Figure 1: The coalescent tree corresponding to the haplotypes 00000, 01000, 01001, 11000, 11100, 11110.

An example of a coalescent tree is given in Figure 1.

Throughout the paper, for a given integer  $k$ , we denote the set  $\{1, \dots, k\}$  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  $B$  that fits the coalescent model such that, for every row  $a$  of  $A$ ,  $B$  contains a pair of rows compatible with  $r$ . 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  $(0, 0), (0, 1), (1, 0), (1, 1)$ . 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 coalescent model.

The assumptions made by the perfect phylogeny model do not necessarily hold in real data, since there are probably back mutations and recombination events. Although in practice these events do happen, it is still reasonable to believe that the behavior of a short region will be close to perfect phylogeny, since it is likely that not too many recombination events or back mutations had occurred throughout history in that region.

## 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 at least one of the sets  $c(A, x) \cap c'(A, y)$ ,  $c(A, x) \cap c'(A, 2)$  or  $c(A, 2) \cap c'(A, y)$  is not empty. 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 (see, e.g., [11]) :

**Lemma 2.1.** *A  $(0, 1)$ -matrix  $B$  is realizable and corresponds to a coalescent tree with root  $(x_1, x_2, \dots, x_m)$  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 root. 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 of the matrix  $A$  (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. Let the resulting matrix be  $A'$ . 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 root, 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  $(c, c')$ ,  $|IND(A, c, c')| \leq 3$ .

### 3 A Simple Algorithm for the PPH Problem

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  $c(A, 2) \cap c'(A, 2) \neq \emptyset$ . 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  $B$ . In order to determine the constraints on a consistent solution we classify the ordered pairs of columns.

Each ordered pair  $(c, c')$  of columns is of one of four types.

**Type A:**  $c(A, 2) \cap c'(A, 2) = \emptyset$ . In this case the pair  $(c, c')$  does not have to be resolved.

In the remaining three cases  $c(A, 2) \cap c'(A, 2) \neq \emptyset$  and the pair  $(c, c')$  does have to be resolved. The cases are as follows:

**Type 0**  $(1, 1) \in IND(A, c, c')$  In this case the pair  $(c, c')$  must be resolved equally.

**Type 1**  $\{(0, 1), (1, 0)\} \subseteq IND(A, c, c')$ . In this case the pair  $(c, c')$  must be resolved unequally.

**Type x**  $(c, c')$  is neither of Type 1 nor of Type 2. In this case  $(c, c')$  may be resolved either equally or unequally.

Note that  $(c, c')$  and  $(c', c)$  are of the same type and must be resolved in the same way (either both equally or both unequally). In completing the description of the algorithm we work with unordered pairs of columns.

A resolution of the pairs of Types 0, 1 and  $x$  can be represented by a symmetric *labeling function*  $L(c, c')$  which is equal to 0 if  $(c, c')$  is resolved equally, and to 1 if  $(c, c')$  is resolved unequally. A labeling function is *legal* if it yields a legal solution to the PPH problem.

For any row  $r$  let  $V_r = \{c \mid A(r, c) = 2\}$ . For a labeling function  $L$ , and a row  $r$ , let  $G_r = (V_r, E_r)$  be the graph with  $V_r$  as the set of vertices and for every  $c, c' \in V_r$ ,  $(c, c') \in E_r$  if and only if  $L(c, c') = 1$ . If  $L, L'$  are two legal labeling functions, we say that  $L'$  *dominates*  $L$ , if  $L(c, c') = 1$  implies that  $L'(c, c') = 1$  and there exist  $(c, c')$  such that  $L'(c, c') = 1, L(c, c') = 0$ .  $L$  is a maximal legal labeling function if there is no  $L'$  which dominates  $L$ .

**Theorem 3.1.** *A labeling function  $L$  is maximal and legal if and only if all of the following holds:*

1. *For every pair  $(c, c')$ ,  $|IND(A, c, c')| \leq 3$ .*
2. *If  $(c, c')$  is of Type 0 then  $L(c, c') = 0$ .*
3. *If  $(c, c')$  is of Type 1 then  $L(c, c') = 1$ .*
4. *For each  $r$ ,  $G_r$  is a complete bipartite graph.*

*Proof.* Assume first that  $L$  is a legal labeling function. Conditions 1, 2, 3 trivially hold. It is easy to see that if there is a path of length  $k$  in  $G_r$  from  $c$  to  $c'$ , then  $L(c, c') = 0$  if  $k$  is even, and  $L(c, c') = 1$  if  $k$  is odd. Thus, there is no odd cycle in  $G_r$ , and therefore  $G_r$  is bipartite. From the above property, it is also easy to see that every connected component of  $G_r$  is a complete bipartite graph. In order to complete the proof we have to show that  $G_r$  is a connected graph. Assume for contradiction that  $C_1$  and  $C_2$  are two different connected components of  $G_r$ . Since both are bipartite, let  $C_1 = R_1 \cup B_1$  and  $C_2 = R_2 \cup B_2$  be the colorings of both components. Let  $L_1$  be defined by  $L_1(c_1, c_2) = 1$  for every  $(c_1, c_2) \in (R_1 \cup R_2) \times (B_1 \cup B_2)$ , and setting  $L_1(c_1, c_2) = L(c_1, c_2)$  for all the other pairs. Let  $L_2$  be defined by  $L_2(c_1, c_2) = 1$  for every  $(c_1, c_2) \in (R_1 \cup B_2) \times (B_1 \cup R_2)$ , and setting  $L_2(c_1, c_2) = L(c_1, c_2)$  for all the other pairs. It is easy to verify that either  $L_1$  or  $L_2$  must be a legal labeling function dominating  $L$ , thus contradicting the maximality of  $L$ .

Assume now that  $L$  satisfies conditions 1, 2, 3, 4. For every row  $r$ , choose some  $c_0$  in  $V_r$ . Replace row  $r$  by two rows  $s$  and  $t$  such that:

1. if  $r(c) \neq 2$  then  $s(c) = t(c) = r(c)$ ;
2.  $s(c_0) = 0$  and  $t(c_0) = 1$ ;
3. if  $c \in V_r$  and  $L(c_0, c) = 0$  then  $s(c) = 0$  and  $t(c) = 1$ ;
4. if  $c \in V_r$  and  $L(c_0, c) = 1$  then  $s(c) = 1$  and  $t(c) = 0$ .

It is easy to verify that, in the resulting  $(0, 1)$ -matrix  $B$ ,  $|IND(B, c, c')| \leq 3$  for all  $(c, c')$ . □

The last condition of this Theorem can be restated as follows. For each row  $r$  choose a *reference column*  $c(r) \in V_r$ . For every pair of columns  $c_1 \in V_r$  and  $c_2 \in V_r$  such that  $c(r), c_1$  and  $c_2$  are distinct,  $L(c_1, c_2) = L(c_1, c_r) + L(c_r, c_2)$ , where addition is modulo 2.

With this restatement we see that all the constraints on a legal labeling function can be expressed as linear equations over  $\text{GF}[2]$  (the field over two elements). The number of variables is at most  $\binom{m}{2}$  and the number of equations is at most  $\frac{nm^2}{2}$ . In polynomial time, using Gaussian elimination, one can either determine that no solution exists or characterize the set of legal solutions in terms of a set of variables that can be chosen freely, such that their values determine the values of the remaining variables.

The polynomial time bound implied by this description is quite high, but in practice many of the pairs will be of Type 1 or Type 0. The values of the corresponding variables are immediately determined, and further variables can be eliminated easily by a forcing process which eliminates a variable whenever it encounters an equation with one or two undetermined variables.

## 4 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,  $(0, 0) \in IND(A, c, c')$  and  $|IND(A, c, c')| \leq 3$ . We define the following relations:

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

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

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

Then for each pair  $(c, c')$  of non-identical columns, 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 coalescent 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 coalescent tree. The only ambiguous case is when  $c \succeq c'$  or  $c' \succeq c$ .

## 4.1 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 repeatedly identifying pairs  $c, c'$  of identical columns in  $A$  such that  $c(A, 1) \cap c'(A, 1) \neq \emptyset$  and deleting one of the two columns. Such identical pairs must be equally resolved, and deleting the repeated copies does not affect the resolution of the remaining pairs of columns. The algorithm then computes the relations between the sites and verifies that, for each pair  $(c, c')$ ,  $(0, 0) \in \text{IND}(A, c, c')$  and  $|\text{IND}(A, c, c')| \leq 3$ .

The algorithm then chooses a *pivot column*  $\hat{c}$  with the property that, for every column  $c$  unequal to  $\hat{c}$ , one of the following holds:  $\hat{c} \succ c$ ,  $\hat{c} \sim c$ , or  $\hat{c} \succeq c$ .

Such a pivot column always exist for the following reason. We can define a relation  $R$  on the columns, where for two columns  $c, c'$ ,  $(c, c') \in R$  if and only if  $c \succ c'$  or  $c \succeq c'$ . One can easily verify that  $R$  is a partial order. If  $\hat{c}$  is a maximal column with respect to  $R$ , then  $\hat{c}$  satisfies the required property. Finding such a maximal column can be done in linear time once the relation between all pairs of columns are stored in the memory (this can be done in the same manner that a maximal number is found in an array).

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  $\hat{c}(A, 2) \cap c(A, 2) \cap d(A, 2) \neq \emptyset$  and  $(1, 1) \in \text{IND}(A, c, d)$ . There is an edge between  $c$  and  $d$  labeled 1 if  $\hat{c}(A, 2) \cap c(A, 2) \cap d(A, 2) \neq \emptyset$  and  $c \sim d$ . There are no other edges except as specified above.

**Lemma 4.1.** *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 4.2.**  *$A'$  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 illustrate the algorithm we provide here a short example. Assume that the given genotypes are  $g_1 = 22200, g_2 = 02220, g_3 = 22000, g_4 = 02020$  and  $g_5 = 01002$ . Let  $c_1, \dots, c_5$  represent the different columns. By definition,  $c_2$  weakly dominates  $c_1, c_3$  and  $c_4$ , and  $c_2$  strongly dominates  $c_5$ . Thus, we choose  $c_2$  to be the pivot. In the graph  $G_{c_2}$ , we have the edges  $(c_1, c_3), (c_3, c_4)$ . The edge  $(c_1, c_3)$  is determined by  $g_1, g_2$  and  $g_3$ , while  $(c_3, c_4)$  is determined by  $g_1, g_2$  and  $g_4$ . The algorithm first constructs the graph, then verifies that it is bipartite. Since it is bipartite, we color it using two colors, red and blue. Assume that  $c_1, c_4$  are blue and  $c_3, c_5$  are red. The algorithm assign two haplotypes  $h_{i1}, h_{i2}$  for every genotype  $g_i$ . The assignment given by the algorithm will be:  $h_{11} = 01100, h_{12} = 10000, h_{21} = 01100, h_{22} = 00010, h_{31} = 01000, h_{32} = 10000, h_{41} = 01000, h_{42} = 00010, h_{51} = 01001$  and  $h_{52} = 01000$ . The algorithm now removes  $c_2$  and the weakly dominated columns and continues recursively with  $c_5$ . Since  $c_5$  is fully determined, the algorithm stops.

In order to implement the algorithm efficiently, we first pre-compute 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  $\hat{c}$ , construct the graph  $G_{\hat{c}}$ , determine how to resolve each pair of sites, and resolve the rows in  $\hat{c}(A, 2)$ . This can be done in time  $O(m^2 | \hat{c}(A, 2) |)$ , and thus the total time for this operation over the course of the algorithm is  $O(m^2n)$ . Hence the running time of the algorithm is  $O(m^2n)$ .

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.

## 5 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. Clearly, when one has a trio, in many cases one can directly infer the haplotype phase from the trios. But in some cases it is impossible to infer it directly. Such a case occurs for SNPs where both the mother, the father and the child are heterozygous. This gives the motivation for an extension of the PPH problem, in which in addition to the coalescent 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:

**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 corresponds to the fact that one of the haplotypes is transmitted from the mother and the other from the father (note that we assume there are no mutations in the transmitted haplotypes, while in the perfect phylogeny model we assume that mutations are allowed throughout history).

### 5.1 Solving TPPH

Recall that the output of algorithm *Build-Tree* is a matrix  $B$  which corresponds to all possible solutions to the coalescent model. For each  $i \in [2n], j \in [m]$ , either  $b_{ij} \in \{0, 1\}$ , or  $b_{ij} \in \{x_1, \dots, x_k, \bar{x}_1, \dots, \bar{x}_k\}$ , 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 5.1.** *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  $x \in \{x_1, \dots, x_k, \bar{x}_1, \dots, \bar{x}_k\}$ , 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  $x \in \{x_1, \dots, x_k, \bar{x}_1, \dots, \bar{x}_k\}$ , and  $x \neq y$ . Any permutations of the values between the mother, father and child does not appear either.

*Proof.* 1. If the  $(t, c)$  configuration is  $(1, 1, *, *, x, \bar{x})$ , then  $c(A, 1) \neq \emptyset, c(A, 2) \neq \emptyset$ . Assume that  $x = x_C$  for some connected component, where  $\hat{c}$  is the maximal site at that point in the algorithm. Then,  $\hat{c} \succeq c$ , and thus,  $c(1) = \emptyset$ , which is a contradiction.

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 1: The possible constraints up to symmetry. The symbol  $z$  represents any value in  $\{0, 1\}$ . The symbol  $x$  represents a literal, that is,  $x \in \{x_1, \dots, x_k, \bar{x}_1, \dots, \bar{x}_k\}$ .

- Since  $x \in \{x_1, \dots, x_k, \bar{x}_1, \dots, \bar{x}_k\}$ , at some point of the algorithm,  $c$  must be in a set  $W_{\hat{c}}$  for some  $\hat{c}$ . At that point all the 2 values of  $c$  will be resolved in the same way, and thus,  $x = y$ . □

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 father 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  $x_1, \dots, x_k$ , 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,  $p(t) = 0$ . 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 1 we list all possible constraints which follow from the  $(t, c)$  configurations. It is easy to verify by Lemma 5.1, 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 1 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 we can express all the constraints in  $C$  by a 2-CNF formula, we can find a legal assignment to the variables by using any of the algorithms known for 2-SAT [1, 5, 7, 19]. If there is no feasible solution, we report that the problem is infeasible.

We note that a simple twist to the above algorithm can also solve the more general problem in which for some children only one of the parents is sequenced. The corresponding constraints should be added to the possible cases of 4-tuples, in the same manner as was done above. The details are omitted since the algorithm and proof are essentially the same as above.

## 6 Minimum Genotype Removal

In practice, the biological data does not exactly fit the coalescent model. We therefore pose the problem of removing the minimum number of individuals from our data set so that the remaining data fits the coalescent model. Formally, we introduce the following problem:

**The Minimum Genotype Removal Problem.** The input is a  $\{0, 1, 2\}$  matrix  $A = (a_{ij})$  of dimensions  $n \times m$ . The goal is to remove the minimum number of rows from  $A$  such that the remaining rows fit the coalescent model.

Clearly, algorithm Build-Tree shows that one can determine in polynomial time if the minimum number of rows is zero. In fact, if the input to the problem is a  $\{0, 1\}$  matrix  $A$ , then the problem can be approximated within a factor of 4 by a local ratio argument. Note that the matrix fits the model if and only if there is no sub-matrix defined by four rows and a pair of columns such that the rows of the sub-matrix contain the pairs  $(1, 1), (1, 0), (0, 1), (0, 1)$ . While there is such a sub-matrix, we simply remove the corresponding four rows. Eventually we will be left with a matrix with no conflict. Since for every four rows that we removed at least one of them should be removed by the optimal solution, we get that this is a 4-approximation to the problem.

### 6.1 The Hardness Result

In this section we show that the minimum genotype removal problem is at least as hard to approximate as the min UnCut problem, which is a well studied optimization problem. In the minimum UnCut problem we are given a graph  $G = (V, E)$ , and we wish to find a minimum size set of edges such that by their removal we are left with a bipartite graph. This problem has a  $\log n$  approximation algorithm [10], where  $n$  is the number of vertices in  $G$ . It is only known to be MAX-SNP hard [20]. We now prove the following theorem:

**Theorem 6.1.** *If the minimum genotype removal problem has an  $\alpha$ -approximation algorithm, then one can find an  $\alpha$ -approximation algorithm for the minimum UnCut problem.*

*Proof.* If  $\alpha > \log n$ , then the theorem trivially holds. We thus assume that  $\alpha \leq \log n$ . Let  $\mathcal{A}$  be an  $\alpha$ -approximation algorithm for the minimum genotype removal problem. Let  $G = (V, E)$  be a graph, where  $V = \{1, \dots, n\}$ ,  $E = \{e_1, \dots, e_m\}$ . We construct the following matrix  $A = (a_{ij})$  as an input to  $\mathcal{A}$ .  $A$  will be of dimension  $m(n\alpha + 1) \times (n + 1)$ . The entries of  $A$  will have the following values:

1. For every  $i \in [m(n\alpha + 1)]$ ,  $a_{i, n+1} = 2$ . (The last column contains only values of 2).
2. For every  $i \in [n]$ ,  $j \in [m\alpha - 1]$ ,  $a_{i, m\alpha - j, i} = 2$ .
3. For every  $i \in [m]$ , if  $e_i = (j, k)$ , then  $a_{i+m\alpha, j} = 2, a_{i+m\alpha, k} = 2$ .
4. Any other entry of  $A$  is zero.

Since the matrix does not contain any entries with 1 values, every two columns must contain a  $(0, 0)$  pair. Therefore, by Lemma 2.1, every sub-matrix of  $A$  which fits the coalescent model, also fits it with the all zeros root. We thus assume that the root is the all zeros root. By property 2, for every  $i \neq j$ ,  $i, j \in [n]$ , we have that  $i$  and  $j$  are siblings. By property 1, column  $n + 1$  is the maximal column. It is easy to see

that the graph  $G_{n+1}$  constructed in algorithm Build-Tree is isomorphic to  $G$ . Each of the last  $m$  rows of  $A$  corresponds to an edge in  $G$ . Furthermore, by removing a subset of the rows of  $A$  so that  $G_{n+1}$  becomes bipartite, we are left with a matrix which fits the coalescent model.

Let  $R$  be the set of rows of  $A$ . Let  $OPT \subseteq E$  be the optimal solution to the min UnCut problem, and let  $OPT' \subseteq R$  be the optimal solution for the minimum haplotype removal problem. Clearly, given a solution  $S \subseteq E$  to the min UnCut problem on  $G$  one can construct a solution to the minimum haplotype removal simply by removing the rows in  $R$  that correspond to  $S$ . Thus,  $|OPT'| \leq |OPT| \leq m$ .  $\mathcal{A}$  returns a set  $S' \subseteq R$  of size at most  $|OPT'|\alpha \leq m\alpha$ . Thus, since every row in  $R$  apart from the last  $m$  rows appears  $m\alpha$  times, we can assume that  $S'$  is a subset of the last  $m$  rows of  $A$ . We now remove from  $G$  every edge which corresponds to a row in  $S'$ , and it is easy to see that the resulting graph is bipartite, and that the number of edges removed is  $|S'| \leq |OPT'|\alpha \leq |OPT|\alpha$ , and thus we get an  $\alpha$ -approximation algorithm to the min UnCut problem.  $\square$

Note that the same proof holds for the problem of removing the minimum number of rows assuming the root is known (in the proof the root is the all zeros vector). To this problem, the local ratio argument gives an approximation ratio of 3 if  $A$  is a  $\{0, 1\}$  matrix.

## 7 Experimental Results

We verified the effectiveness of our technique over a set of data presented in Daly et al., 2001 [6]. In their study, they collected the genotypes for 103 SNPs in a 500 kilobase region of chromosome 5p31 from 129 mother, father, child trios. Using these trios, they inferred the haplotypes for each individual using Mendelian heredity. 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 common haplotypes account for 90% of the individual chromosomes.

We tested our method by predicting the haplotypes of the children in the trios only given their genotypes. Using the haplotypes inferred from the trios, we were able to verify the accuracy of our predictions. For each of the 11 blocks defined by Daly et. al., 2001 [6], the predictions of Build-Tree are shown in Table 2. In all cases, the algorithm chooses the correct common haplotypes and the overall error rate is below 1%. Thus, using our algorithm, the study of Daly et al., 2001 [6] could have been done by collecting the genotypes from *unrelated* individuals instead of collecting data from trios.

Several issues arise in practice when applying the algorithm to real data. The first is that typically a large proportion of the genotype data is missing. In the Daly et al., 2001 [6] the proportion of missing genotype data is 10.03%. A second issue is that the actual haplotypes typically do not perfectly fit the perfect phylogeny model, especially the infrequent haplotypes. However, in practice, the haplotypes roughly fit the perfect phylogeny model. Heuristics to handle both missing data and conflicts with the perfect phylogeny model are described in detail in [15] where we present experimental results of the technique in more depth.

## 8 Concluding Remarks

We presented a practical and efficient algorithm to infer haplotype structure from genotype data. We furthermore extended our algorithm to cope with further constraints. We presented relations between classical combinatorial problems and the suggested biological problem. Our experiments show that our algorithms are practical, and could save time and money in biological experiments. We believe that further extensions to this problem could lead to even more accurate haplotype reconstruction on a larger scale. Specifically, coping with errors in the data and with missing data is left as an open problem.

SNPs	Actual Common Haplotypes	Predicted Common Haplotypes	Frequency	Error Rate
1-8	GGACAACC AATTCGTG	GGACAACC AATTCGTG	215 38	0
10-14	TTACG CCCAA	TTACG CCCAA	217 35	0
16-24	CGGAGACGA GACTGGTCG CGCAGACGA	CGGAGACGA GACTGGTCG CGCAGACGA CGGATACGA	139 52 34 15	0.005780
25-35	CGCGCCCGGAT CTGCTATAACC TTGCCCCGGCT* CTGCCCCAACC*	CGCGCCCGGAT CTGCTATAACC CTGCCCCGGCT TTGCCCAACC	142 39 35 25	0
36-40	CCAGC CCACC GCGCT CAACC	CCAGC CCACC GCGCT CAACC	146 51 30 12	0
41-45	CCGAT CTGAC ATACT	CCGAT CTGAC ATACT	152 63 31	0.011561
78-84	CGTTTAG TGTT*GA TGATTAG CGTCTAG	CGTTTAG TGTTTGA TGATTAG CGTCTAG TGTTGGA	142 53 20 12 10	0
86-91	ACAACA GCGGTG ACGGTG GTGACG	ACAACA GCGGTG ACGGTG GTGACG	145 71 14 13	0.007353
92-98	GTTCTGA TGTGTAA TG*GCGG	GTTCTGA TGTGTAA TGTGCGG TGCGTAA	142 49 32 15	0.004132
99-103	CGGCG TATAG TATCA	CGGCG TATAG TATCA	112 105 35	0.003448

Table 2: Predictions over data from Daly et al. 2001, [6]. The second column shows the common haplotypes as presented in Daly et al. 2001. The third column shows the predictions and the fourth gives their frequencies. The fifth column shows the error ratio in our predictions. The error rate is the fraction of bases the algorithm guessed wrong over the whole data set, and not only over the individuals with the common haplotypes.

## References

- [1] B Aspval, MF Plass, and RE 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 UC Davis CSE-2002-21*, Jul 17 2002.
- [3] RE Bixby and DK Wagner. An almost linear time algorithm for graph realization. *Mathematics of Operations Research*, 13:99–123, 1988.
- [4] AG Clark. Inference of haplotypes from pcr-amplified samples of diploid populations. *Journal of Molecular Biology and Evolution*, 7(2):111–22, Mar 1990.
- [5] SA Cook. The complexity of theorem-proving procedures. *STOC*, pages 151–158, 1971.
- [6] 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.
- [7] S Even, A Itai, and A Shamir. On the complexity of timetable and multicommodity flow problems. *SICOMP*, 5:691–703, 1976.
- [8] 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.
- [9] 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.

- [10] N Garg, VV Vazirani, and M Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM J. Comp.*, 25:235–251, 1996.
- [11] D Gusfield. Efficient algorithms for inferring evolutionary history. *Networks*, 21:19–28, 1991.
- [12] 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.
- [13] D Gusfield. Inference of haplotypes from samples of diploid populations: complexity and algorithms. *Journal of Computational Biology*, 8(3):305–23, 2001.
- [14] D 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.
- [15] E Halperin and E Eskin. A practical solution for haplotype mapping. *Unpublished Manuscript*, 2002.
- [16] 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.
- [17] B 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.
- [18] 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.
- [19] CH Papadimitriou. On selecting a satisfying truth assignment. *FOCS*, pages 163–169, 1991.
- [20] CH Papadimitriou and M Yannakakis. Optimization, approximation and complexity classes. *JCSS*, 43:425–440, 1991.
- [21] 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 Trulsson, 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.
- [22] 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.
- [23] WT Tutte. An algorithm for determining whether a given binary matroid is graphic. *Proc. of Amer. Math. Soc.*, 11:905–917, 1960.