UC Davis Computer Science Technical Report CSE-2002-21

Haplotyping as Perfect Phylogeny: A direct approach

Vineet Bafna and Dan Gusfield and Giuseppe Lancia and Shibu Yooseph

July 17, 2002

Haplotyping as Perfect Phylogeny: A direct approach

Vineet Bafna[*] Dan Gusfield[†] Giuseppe Lancia[‡] Shibu Yooseph[§]

July 17, 2002

Abstract

A full *Haplotype Map* of the human genome will prove extremely valuable as it will be used in large-scale screens of populations to associate specific haplotypes with specific complex genetic-influenced diseases. A prototype Haplotype Mapping strategy is presently being finalized by an NIH working group. The biological key to that strategy is the surprising fact that genomic DNA can be partitioned into long blocks where genetic recombination has been rare, leading to strikingly fewer distinct haplotypes in the population than previously expected [10, 3, 12, 4].

In this paper we explore the algorithmic implications of the *no-recombination in long blocks* observation, for the problem of inferring haplotypes in populations. This assumption, together with the standard population-genetic assumption of infinite sites, justifies a model of haplotype evolution where the haplotypes in a population are assumed to evolve along a coalescent, which as a rooted tree is a perfect phylogeny. We consider the following algorithmic problem, called Perfect Phylogeny Haplotyping problem (PPH), which was introduced by Gusfield [9] - given n genotypes, does there exist a set of at most $2n$ haplotypes such that each genotype is generated by a pair of haplotypes from this set, and such that this set can be derived on a perfect phylogeny? The approach taken in [9] to solve this problem reduces it to established, very deep, results and algorithms from matroid and graph theory. Although that reduction is quite simple, and the resulting algorithm nearly optimal in speed (a linear time lower bound is necessary), taken as a whole that approach is quite involved, and in particular, challenging to program. Moreover, anyone wishing to fully establish, by reading existing literature, the correctness of the entire algorithm would need to read several deep and difficult papers in graph and matroid theory. A stated open goal in [9] was to find a simpler more direct, yet still efficient, algorithm, via a self-contained approach (not needing deep matroid or graph theorems). This paper accomplishes that goal, for both the rooted and unrooted PPH problems. It establishes a very simple, easy to program, $O(nm^2)$-time algorithm that determines whether there is a PPH solution for input genotypes, and produces a linear-space data-structure to represent all of the solutions. The approach allows complete, although not trivial, self-contained proofs. In addition to algorithmic simplicity, the approach here makes the representation of all solutions more intuitive than in [9], and solves another open goal from that paper, namely to prove a non-trivial upper bound on the number of PPH solutions, showing that that number is vastly smaller than the number of haplotype solutions (each solution being a set of n pairs of haplotypes that can generate the genotypes) when the perfect phylogeny requirement is not imposed.

1 Introduction

The next high-priority phase of human genomics will involve the development of a full *Haplotype Map* of the human genome [10]. It will be used in large-scale screens of populations to associate

[*]Informatics Research, Celera, 45 W. Gude Drive, Rockville, MD 20850. Email: Vineet.Bafna@celera.com
[†]Dept. of Computer Science, 3051 Engineering II, University of California, One Shields Avenue, Davis, CA 95616. Email: gusfield@cs.ucdavis.edu Research Supported by NSF grant DBI-9723346
[‡]DEI, University of Padova, Via Gradenigo, 6-A, 35131 Padova, Italy. Email: lancia@dei.unipd.it
[§]Informatics Research, Celera, 45 W. Gude Drive, Rockville, MD 20850. Email: Shibu.Yooseph@celera.com

specific haplotypes with specific complex genetic-influenced diseases. A prototype Haplotype Mapping strategy is presently being finalized by an NIH working-group. The biological key to that strategy is the surprising fact that genomic DNA can be partitioned into long blocks where genetic recombination has been rare, leading to strikingly fewer distinct haplotypes in the population than previously expected [10, 3, 12, 4].

Mathematically, this lack of recombination (along with the infinite sites assumption) justifies a model of haplotype evolution where the haplotypes in a population are assumed to evolve along a coalescent, which as a rooted tree is a perfect phylogeny [14, 9]. This leads to the algorithmic problem of determining whether unphased diploid genotype data is consistent with an evolutionary history on a tree, and to find such a tree if one exists. That is, we must find for each of the n given genotypes, a pair of haplotypes (binary vectors) which generate that genotype, so that the entire set of $2n$ haplotypes can be derived on a perfect phylogeny.

Given haplotypes (phased data), it is easy to determine if the haplotypes could have evolved along a perfect phylogeny. The difficult, and currently essential problem is to determine whether unphased (i.e. genotype) data could have evolved along a perfect phylogeny. This is called the Perfect Phylogeny Haplotyping problem (PPH problem).

The ability to solve the PPH problem is of value for many purposes, notably the ability to identify in genotype data, blocks of DNA-SNP data where no recombination has occurred.

Gusfield [9] showed that the PPH problem can be solved in almost-linear time; that in linear additional time it can be determined if the solution is unique; and if not, then in linear additional time, one can build a linear-space data structure that represents all the solutions, so that each can be generated in linear time. That paper also showed how to find at least one solution to an unrooted version of the problem, or determine that it has no solutions. The approach in that paper reduces the PPH problem to established, very deep, results and algorithms from matroid and graph theory. Although that reduction is quite simple, and the resulting algorithm nearly optimal in speed (a linear time lower bound is necessary), taken as a whole that approach is quite involved, and in particular, challenging to program. Moreover, anyone wishing to fully establish, by reading existing literature, the correctness of the entire algorithm would need to read several deep and difficult papers in graph and matroid theory.

A stated open goal in [9] was to find a simpler more direct, yet still efficient, algorithm, via a self-contained approach (not needing deep matroid or graph theorems). This paper accomplishes that goal, for both the rooted and unrooted PPH problems. It establishes a very simple, easy to program, $O(nm^2)$-time algorithm that determines whether there is a PPH solution for input genotypes, and produces a linear-space data-structure to represent all of the solutions. The approach allows complete, although not trivial, self-contained proofs. In addition to algorithmic simplicity, the approach here makes the representation of all solutions more intuitive than in [9], and solves another open goal from that paper, namely to prove a non-trivial upper bound on the number of PPH solutions, showing that that number is vastly smaller than the number of haplotype solutions (each solution being a set of n pairs of haplotypes that can generate the genotypes) when the perfect phylogeny requirement is not imposed.

1.1 Introduction to SNP's, Genotypes and Haplotypes

In diploid organisms (such as humans) there are two (not completely identical) "copies" of each chromosome, and hence of each region of interest. A description of the data from a single copy is called a *haplotype*, while a description of the conflated (mixed) data on the two copies is called a *genotype*. In complex diseases (those affected by more than a single gene) it is often much more informative to have haplotype data (identifying a set of gene alleles inherited together) than to have only genotype data.

The underlying data that forms a haplotype is either the full DNA sequence in the region, or more commonly the values of *single nucleotide polymorphisms (SNP's)* in that region. A SNP is a single nucleotide site where exactly two (of four) different nucleotides occur in a large percentage of the population. The SNP-based approach is the dominant one, and high density SNP maps have been constructed across the human genome with a density of about one SNP per thousand nucleotides.

1.2 The biological problem

Because polymorphism screens will be conducted on large populations, it is not feasible to examine the two copies of a chromosome separately, and *genotype* data rather than haplotype data will be obtained, even though it is the haplotype data that will be of greatest use.

Abstractly, data from m sites (SNP's) in n individuals is collected, where each site can have one of two states (alleles), which we denote by 0 and 1. For each individual, we would ideally like to describe the states of the m sites on each of the two chromosome copies separately, i.e., the haplotype. However, experimentally determining the haplotype pair is technically difficult or expensive. Instead, the screen will learn the $2m$ states (the genotype) possessed by the individual, without learning the two desired haplotypes for that individual. One then uses computation to extract haplotype information from the given genotype information. Several methods have been explored and are intensely used for this task [2, 1, 5, 13, 8, 11]. None of these methods are presently fully satisfactory.

1.3 The Haplotype Inference (HI) problem

Given n genotype vectors, the HI problem is to find n pairs of haplotype vectors that could have generated the genotype vectors.

More formally, we are given an n by m genotype matrix M with $M[i,j] \in \{0,1,2\}$. The i-th row $M[i,*]$ describes the genotype of species s_i, and each location j where $M[i,j]=2$ is a polymorphic site. Each column $c_j = M[*,j]$ is a polymorphic locus. The goal is to generate a $2n \times m$ *haplotype-matrix* M', with $M'[i,j] \in \{0,1\}$. A $2n \times m$ haplotype-matrix M' is an *expansion* of a $n \times m$ genotype matrix M, if the following hold.

1. Each row $M[i,*]$ expands to two rows denoted by $M'[i,*]$, and $M'[i',*]$.

2. For all j s.t. $M[i,j] \in \{0,1\}$, $M[i,j] = M'[i,j] = M'[i',j]$.

3. For all j s.t. $M'[i,j] = 2$, $M'[i,j] \neq M'[i',j]$.

Any such M' is a feasible explanation for the origin of M. However, if each i has h_i polymorphic sites, then there are $\Pi_{i=1}^{i=n} 2^{h_i - 1}$ solutions to the HI problem. Of course, given M we want to find the "true" haplotype matrix M' which originally gave rise to M. That goal would be impossible without the implicit or explicit use of some genetic model, either to assess the biological fidelity of any proposed solution, or to guide the algorithm in constructing a solution.

In this paper, we consider the genetic model where a solution to the HI problem (the n pairs of binary vectors) is required to determine a *perfect phylogeny.* Below is a formal definition of a (rooted) perfect phylogeny.

Definition Let M be an $2n$ by m 0-1 (binary) matrix. Let V be an m-length binary vector, called the *ancestor vector* (V is often assumed, without loss of generality, to be the all-0 vector.)

A *perfect phylogeny for M and V* is a rooted tree T with exactly $2n$ leaves that obeys the following properties:

1) Each of the $2n$ rows labels exactly one leaf of T, and each leaf is labeled by one row.

2) Each of the m columns labels *exactly one* edge of T.

3) Every interior edge (one not touching a leaf) of T is labeled by *at least* one column.

4) For any row i, the value $M(i,j)$ is unequal to $V(j)$ if and only if j labels an edge on the unique path from the root to the leaf labeled i. Hence, that path is a compact representation of row i.

The biological interpretation is that an edge label j indicates the point in time where a mutation at site j occurred, and so the state of site j changes from its ancestral value to the opposite value. The justification for the perfect phylogeny model is based on recent observations of no or little recombination in long segments of DNA, and the standard infinite-sites assumption. See [14, 9] for a full justification of this model.

In the *rooted* version of perfect phylogeny, V is given as input. There is also an *unrooted* version of perfect phylogeny, where V is not specified. In that case, a binary matrix M is said to have a perfect phylogeny if *there exists* a V such that there is a (rooted) perfect phylogeny for M and V.

Formally, the *Perfect Phylogeny Haplotype (PPH) Problem* is: Given M, find an expansion M' of M which defines an unrooted perfect phylogeny.

This definition is slightly different from the one given in [9]. There the problem was to find an expansion M' which defines a rooted perfect phylogeny, assuming the ancestor vector V is already known. However, the rooted and the unrooted PPH problems are equivalent, in the sense that an algorithm for one version can be used to solve either version. For example, with an algorithm for the unrooted version, we solve an instance of the rooted version as follows: Given both M and V, simply add a genotype row v consisting of vector V to M; there will be a leaf labeled with v in the resulting perfect phylogeny for this M; a rooted perfect phylogeny for M and V is then obtained by making v the root. The unrooted version can also be reduced to the rooted version, but we will not need that direction in this paper.

The solution to the perfect phylogeny problem given in [9] is based on (complex) graph theoretic tools. Although perfect phylogeny is defined in terms of trees, there is an alternative characterization that allows one to focus more on the matrix, and that is the approach taken in this paper.

2 An alternative characterization of the PPH problem

The following characterization of perfect phylogeny has been independently established many times, and is described in many places. See [6, 7] for one such exposition.

Define a *complete-pair-matrix* as matrix with 2 columns, containing each of the rows in $\{00, 01, 10, 11\}$. The classical theorem is that a $2n \times m$ binary matrix M' defines a unrooted perfect phylogeny if and only if no submatrix $M'[*, (j_1, j_2)]$ formed by selecting the two columns j_1, j_2, is a *complete-pair-matrix.*

In these terms, the PPH problem is to find an expansion M' of M which does not contain a complete-pair-matrix. This definition of the PPH problem avoids any explicit mention of trees and allows a more matrix-oriented solution. We now begin to develop the tools needed for that solution.

Proposition 1: Consider a sub-matrix $M[*, (j_1, j_2)]$ of a genotype matrix M.

1. All rows not of the form 22 have a unique expansion in any haplotype matrix M'.
2. If this expansion contains a complete-pair-matrix, no PPH is possible for M.

Definition: We denote a pair of columns j_1, j_2 as a *pph*-pair if the non 22 rows in $M[*, (j_1, j_2)]$ do not expand to include a complete-pair-matrix.

In (binary) haplotype data, blocks of SNP/DNA where recombination has not occurred are frequently identified by checking for the absence of a complete-pair-matrix. For example see [2, 14]. This is called the "four-gamate test" in the biological literature. The ability to solve the PPH problem is of great value because it allows one to identify these no-recombination blocks using genotype data instead of haplotype data.

Unlike the problem of inferring a Perfect Phylogeny on a binary matrix, we note that there are no simple characterizations of the existence of a solution to the PPH problem. We might ask, for instance, does every pair of columns being *pph*-pair guarantee the existence of a solution? Or, if

	j_1	j_2	j_3	j_4
i_1	2	0	2	2
i_2	0	2	2	2
i_3	2	2	0	2

Figure 1: Consider the above instance of the PPH problem with rows i_1, i_2, i_3 and columns j_1, j_2, j_3, j_4. This instance is such that every pair of columns is a pph-pair; furthermore, every triple of columns can be expanded so that the result does not contain a complete submatrix. However, this PPH instance does not have a solution.

every triple of columns can be expanded so that the triple has no complete submatrix, then does the PPH instance have a solution? The example in Figure 1 shows that both of these conjectures are false.

Definition: Columns j_1, j_2 in a genotype matrix M are *companions*, if there exists a row i such that $M[i,j_1] = M[i,j_2] = 2$. Row i is a *companion* row for j_1, and j_2. Let M' be a PPH of M, and i be a companion row for columns j_1, j_2. Row i *equates* j_1 and j_2 in M' if $M'[i,j_1] = M'[i,j_2]$ (and, consequently $M'[i',j_1] = M'[i',j_2]$). Otherwise, row i *negates* j_1, and j_2. Given M', define an indicator function $\mathcal{E}$ on companion rows and companion columns as follows - $\mathcal{E}(i,j_1,j_2) = 0$ if companion row i equates companion columns j_1, and j_2, and $\mathcal{E}(i,j_1,j_2) = 1$ if companion row i negates companion columns j_1 and j_2.

From the above definition of $\mathcal{E}$ it follows that, for row i that is a companion row for companion columns j_1 and j_2, $\mathcal{E}(i,j_1,j_2) = M'[i,j_1] \oplus M'[i,j_2]$, where $\oplus$ denotes the EXCLUSIVE-OR operator; recall that for boolean values a and b, $a \oplus b = 1$ iff $a \neq b$.

Lemma 2: Consider a set of columns $J = \{j_1, j_2, \ldots, j_l\}$ of a genotype matrix M, with a companion row i (i.e. $M[i,j_1] = M[i,j_2] = \ldots = M[i,j_l] = 2$). Let M' be a solution to the PPH problem. Then

$$\mathcal{E}(i,j_x,j_z) = \mathcal{E}(i,j_x,j_y) \oplus \mathcal{E}(i,j_y,j_z) \quad \forall j_x, j_y, j_z \in J$$

Proof: As $M[i,j_x] = M[i,j_y] = M[i,j_z] = 2$, we have that $M'[i,j_x] = \mathcal{E}(i,j_x,j_y) \oplus M'[i,j_y]$, and $M'[i,j_z] = M'[i,j_y] \oplus \mathcal{E}(i,j_y,j_z)$. Also by definition,$\mathcal{E}(i,j_x,j_z) = M'[i,j_x] \oplus M'[i,j_z]$. Expanding, $\mathcal{E}(i,j_x,j_z) = \mathcal{E}(i,j_x,j_y) \oplus \mathcal{E}(i,j_y,j_z)$. ♣

Lemma 3: If rows i_1 and i_2 are companion rows for columns j_1, j_2, then $\mathcal{E}(i_1,j_1,j_2) = \mathcal{E}(i_2,j_1,j_2)$.

Proof: Assume to the contrary. Then, the sub-matrix of M' formed by rows i_1, i'_1, i_2, i'_2, and columns j_1, j_2 is a *complete-pair-matrix*, a contradiction. ♣

Therefore, in the following, we can speak of companion columns j_1, j_2 being equated, or negated, without reference to a row. Sometimes, two columns are *forced* to be equated or negated to avoid the complete pair-matrix. For companion columns j_1, j_2, consider the unique expansion of the non 22 rows in $M[*, (j_1, j_2)]$. If this expansion contains the rows 00 and 11, then columns j_1 and j_2 *must* be equated in any PPH M'. Likewise, if the expansion contains the rows 01 and 10, columns j_1 and j_2 *must* be negated in any PPH M'. Two rows $\{M[i_1,(j_1,j_2)], M[i_2,(j_1,j_2)]\}$ form a *forcing-pattern* for columns j_1, j_2, if they do not contain 22, and their unique expansion either contains both 00, and 11, or 01, and 10. The following lemma will be used extensively later.

Lemma 4: Consider companion columns j_1, j_2 in M. Then, $\{x2, 2y\}$ is a forcing-pattern for $M[*, (j_1, j_2)]$ for all $x, y \in \{0, 1\}$.

Note that any M' that solves the PPH problem for M also defines a single corresponding indicator function $\mathcal{E}(j_1, j_2)$ for all companion columns j_1, j_2. It is easy to establish that two distinct solutions to an instance of the PPH problem produce two distinct indicator functions.

We now present the major theorem that is the organizing idea behind the PPH solution developed in this paper.

Theorem 5: Consider a genotype matrix M, such that every pair of columns is a pph-pair. There exists a solution M' to the PPH problem if and only if there is a 0-1 valued indicator function $\mathcal{E}$ defined on companion columns j_1, j_2 with the following properties:

1. If companion columns j_1, j_2 have a forcing pattern that equates j_1 and j_2, then $\mathcal{E}(j_1, j_2) = 0$. If the forcing pattern negates j_1, and j_2, then $\mathcal{E}(j_1, j_2) = 1$.
2. $\mathcal{E}(j_x, j_z) = \mathcal{E}(j_x, j_y) \oplus \mathcal{E}(j_y, j_z)$ $\forall j_x, j_y, j_z$ s.t. $M[i, j_x] = M[i, j_y] = M[i, j_z] = 2$ for some i.

Proof: Lemmas 2 and 3 show that if a PPH solution exists, then we can define an indicator function that satisfies the stated two properties.

Now, suppose there exists an indicator function $\mathcal{E}$ defined on companion columns that has the two stated properties. We will show that we can derive a solution to the PPH problem using $\mathcal{E}$. Consider the haplotype matrix M' constructed according to Algorithm $\mathcal{E}2M$, which is described in Figure 2. This algorithm uses the properties of the indicator function $\mathcal{E}$ to set the values of entries in M'. Clearly, the haplotype matrix M' is an expansion of the genotype matrix M.

We next establish the following consistency claim (which also shows that the choice of index k in step b of the algorithm has no effect on the resulting expansion): For any column j, if there exists companion columns k and k' where $k' \leq k < j$, such that $M[i, k'] = M[i, k] = M[i, j] = 2$, then $M'[i, k'] \oplus \mathcal{E}(k', j) = M'[i, k] \oplus \mathcal{E}(k, j)$. The claim is trivially true for $j = 2$. Assume it is true up to some value of j, and consider the next j. Again, if there is at most one index $l < j$ where $M[i, l] = M[i, j] = 2$, then the claim is trivially true. Otherwise, consider indices $k' < k < j$ where $M[i, k'] = M[i, k] = M[i, j] = 2$, By induction, $M'[i, k] = M'[i, k'] \oplus \mathcal{E}(k', k)$ (even if k' is not the column used by the algorithm when setting $M'[i, k]$), and so using a property of the $\oplus$ operator, $\mathcal{E}(k', k) = M'[i, k] \oplus M'[i, k']$. Now the consistency claim, $M'[i, k'] \oplus \mathcal{E}(k', j) = M'[i, k] \oplus \mathcal{E}(k, j)$, is equivalent to $(M'[i, k'] \oplus \mathcal{E}(k', j)) \oplus (M'[i, k] \oplus \mathcal{E}(k, j)) = 0$, which is what we will prove. Observe that

$$\begin{array}{cll}
 & (M'[i,k'] \oplus \mathcal{E}(k',j)) \oplus (M'[i,k] \oplus \mathcal{E}(k,j)) & \\
= & (M'[i,k'] \oplus M'[i,k]) \oplus (\mathcal{E}(k',j) \oplus \mathcal{E}(k,j)) & \\
= & (\mathcal{E}(k,k')) \oplus (\mathcal{E}(k',j) \oplus \mathcal{E}(k,j))) & \text{(from above)} \\
= & (\mathcal{E}(k,k')) \oplus (\mathcal{E}(k',k)) & \text{(from property 2 of the Theorem statement)} \\
= & 0 &
\end{array}$$

We now use this consistency to show that M' is a solution to the PPH problem by showing that it does not contain a complete-pair submatrix. Assume it does contain one, denoted S, and let j_1, j_2, where $j_1 < j_2$, be the columns of S.

Let R (possibly empty) be the submatrix of S consisting of the rows of S which were expanded from companion rows for j_1, j_2. If R is empty, then S contains a complete submatrix only if columns j_1 and j_2 are not pph, contradicting the condition of the theorem. So we assume that R is not empty. Let $A = \{00, 11\}$ and $B = \{01, 10\}$. By the consistency proved above, either every row of R is from set A, or every row of R is from set B. Without loss of generality, assume that the rows of R are from set A (the case when they are from B is symmetric). That means that $\mathcal{E}(j_1, j_2) = 0$.

Now since the rows of R cannot contain the rows specified in B, if S contains a complete submatrix, there must be two rows of S not in R which contain exactly the two rows in B. But since these rows are not companion rows for j_1 and j_2, the rows will contain $\{01, 10\}$ in every expansion of j_1, j_2. Hence j_1 and j_2 are forced to be negated in $\mathcal{E}$, i.e., they force $\mathcal{E}(j_1, j_2) = 1$, a contradiction.

```
Input: A genotype matrix $M$, and indicator function $\mathcal{E}$ defined on companion columns

Output: A haplotype matrix $M'$ which is an expansion of $M$

Algorithm $\mathcal{E}2M$:
    For each of the $m$ columns $j, 1 \leq j \leq m$ in turn, do the following

        1. For each row $i$,
            (a) if $M[i,j] = x \in \{0,1\}$, set $M'[i,j] = M'[i',j] = x$.
            (b) Otherwise, if $\exists k < j$ such that $M[i,l] = 2$, let $k < j$ be any index less than $j$ such
                that $M[i,k] = 2$, and set $M'[i,j] = M'[i,k] \oplus \mathcal{E}(j,k)$, and $M'[i',j] = 1 - M[i,j]$.
            (c) If no such index $k$ exists, set $M'[i,j] = 1$, and $M'[i',j] = 0$.
```

Figure 2: Algorithm $\mathcal{E}2M$ takes an indicator function and a genotype matrix M as input, and produces a haplotype matrix M'

Therefore $\mathcal{E}$ can be used to construct a haplotype matrix M' from M.

♣

Theorem 5 gives us an alternative characterization of the PPH problem in terms of setting an appropriate indicator function. Note that the indicator function is only defined on companion column pairs.

In addition to finding one PPH solution, we want efficient methods to find and count all the solutions. For that purpose, we need the following Lemma.

Lemma 6: Let M'_1 and M'_2 be solutions to the PPH problem for genotype matrix M. Furthermore, let M'_1 be generated by the indicator function $\mathcal{E}_1$ and M'_2 be generated by the indicator function $\mathcal{E}_2$. Then $M'_1 = M'_2$ if $\mathcal{E}_1 = \mathcal{E}_2$.

Proof: Suppose $M'_1 \neq M'_2$. Recall that any submatrix $M[i,(j_1,j_2)]$ where $M[i,j_1]$ and $M[i,j_2]$ are not both equal to 2, has a unique expansion in any haplotype matrix. Thus, $M'_1 \neq M'_2$ implies that there is a row i that is a companion row for columns j_1 and j_2 and for which the submatrices in M'_1 and M'_2, that are generated by the expansion of the submatrix $M[i,(j_1,j_2)]$, are different. Since $M[i,j_1] = M[i,j_2] = 2$, these submatrices will be $[(00)(11)]$ in one (say M'_1) and $[(01)(10)]$ in the other (say M'_2). Thus, in M'_1, we have that $\mathcal{E}_1(j_1,j_2) = 0$ and in M'_2, we have that $\mathcal{E}_2(j_1,j_2) = 1$. This implies that $\mathcal{E}_1 \neq \mathcal{E}_2$. ♣

The converse of this Lemma is also true, but will not be needed.

We can see at this point that if M contains a column j with no 2 entry, then column j can be removed since it is not in any companion pair. There will be a PPH solution for M if and only if there is a PPH solution on the remaining columns, and all pairs of columns in M are pph-pairs. Hence, we assume from this point that every column contains at least one 2 entry.

3 An Algorithm for the PPH problem: Connected Components

Definition: For a genotype matrix M, define an associated *genotype* graph $G_M(J, E_f \cup E_n)$ (abbreviated G) as follows: The vertex set $J = \{j_1, \ldots, j_n\}$ is the set of columns in M. There are two sets of edges. A column pair $(j,j') \in E_f$ if and only if there is a forcing pattern in $M[*,j,j']$ and j,j'

	j_1	j	j'	j_2
i_1	2	2	x	
i_2		y	2	2
i		2	2	

Figure 3: A submatrix describing a path j_1, j, j', j_2 in $G_M(J, E_f \cup E_n)$ containing $(j, j') \in E_n$. Either $(j_1, j') \in E_f \cup E_n$ or $(j, j_2) \in E_f \cup E_n$.

are companion columns. Correspondingly, a column pair $(j, j') \in E_n$ if j, j' are companion columns, but $M[*, j, j']$ does not have a forcing pattern. We use $G_M(J, E_f)$ to denote the graph with the same sets of nodes, but with edge set restricted to E_f.

In the following, we will show that the edges from E_n have many restricting properties.

Lemma 7: Consider a triangle in $G_M = (J, E_f \cup E_n)$ formed by columns j_1, j_2, j_3 in M. If the edge formed by any pair is in E_n, there exists a row i with $M[i, j_1] = M[i, j_2] = M[i, j_3] = 2$.

Proof: Assume otherwise. W.l.o.g, let $(j_1, j_3) \in E_n$. Since all pairs are companions (by definition), there must exist rows i_1, i_2, i_3 with the following properties:

1. $M[i_1, j_1] = M[i_1, j_2] = 2 \neq M[i_1, j_3]$
2. $M[i_2, j_3] = M[i_2, j_2] = 2 \neq M[i_1, j_1]$
3. $M[i_3, j_1] = M[i_2, j_3] = 2 \neq M[i_1, j_2]$

Then, from Lemma 4, the submatrix $M[*, (j_1, j_3)]$ contains the forcing pattern $\{x2, 2y\}$, where $x, y \in \{0, 1\}$, a contradiction ♣

The following is the key theorem describing the edges from E_n in G_M. It says that while G_M can have cycles of arbitrary length containing only edges from E_f, cycles of length greater than 3 containing edges from E_n, must have chords in them.

Theorem 8: (Weak Triangulation) In $G_M(J, E_f \cup E_n)$, every cycle of length greater than 3 that has an edge from E_n, contains a chord.

Proof: Assume to the contrary. Let $(j, j') \in E_n$ be an edge in a chordless cycle of length $k > 3$. Let j_1 be adjacent to j, and j_2 be adjacent to j' in the cycle. As $(j_1, j), (j, j'), (j', j_2)$ are companion columns, there exist rows i_1 such that $M[i_1, j_1] = M[i_1, j] = 2$, i_2 such that $M[i_2, j'] = M[i_2, j_2] = 2$, and i such that $M[i, j] = M[i, j'] = 2$. Note that $x \neq 2$ as there is no edge (chord) between j_1 and j'. Likewise $y \neq 2$. See figure 3. Thus $M[*, j, j']$ contains the pattern $\{x2, 2y\}$ with $x, y \in \{0, 1\}$. By Lemma 4, $(j, j') \in E_f$, a contradiction. ♣

Corollary 9: In $G_M(J, E_f \cup E_n)$, consider a cycle of length 3 or more that contains nodes $j_1, j_2, ..., j_k$. Let edge $(j_1, j_k) \in E_n$. Then, either $(j_1, j_{k-1}) \in E_f \cup E_n$, or $(j_2, j_k) \in E_f \cup E_n$.

Theorem 10: Consider a genotype matrix M which has at least one PPH solution. Let M' be an any of the PPH solutions for M, and $\mathcal{E}_{\mathcal{M}'}$ be the indicator function defined by M'. Let C be any single connected component of $G_M(J, E_f)$. Then Algorithm *PPH-CC* (described in Figure 4) sets an $\mathcal{E}$ value for each edge in C, and those values are precisely the values of $\mathcal{E}_{\mathcal{M}'}$.

Proof: We first show that the algorithm sets an $\mathcal{E}$ value for each edge in C. Clearly, it does so for each edge in E_f. Next we show that as long as $\mathcal{E}$ is not set for all edges in E_n, there exists a triple

```
Algorithm PPH-CC:
Input: An $n \times m$ genotype matrix M, where all pairs are pph, and a single connected component $C$
    of $G(J, E_f)$.
Output: 0-1 valued functions $\mathcal{E}$ defined on each companion pair of columns $j_1, j_2$ whose associated
    nodes are in $C$.
Algorithm:
    Set:
        1. For each edge $\mathcal{E}(j_2, j_2) \in C$, set $\mathcal{E}(j_1, j_2) = 0$ if there is a forcing pattern that equates
           $j_1, j_2$. Set $\mathcal{E}(j_1, j_2) = 1$ if there is a forcing pattern that negates $j_1, j_2$.
        2. Until unable, iteratively find three columns $j_1, j_2, j_3$ in $C$ such that $\mathcal{E}(j_1, j_2)$, and
           $\mathcal{E}(j_2, j_3)$ are set, but $\mathcal{E}(j_1, j_3)$ is not and $j_1, j_3$ are companions; set $\mathcal{E}(j_1, j_3) = \mathcal{E}(j_1, j_2) \oplus$
           $\mathcal{E}(j_1, j_2)$
    Output: Output $\mathcal{E}$ for the edges in $C$
```

Figure 4: Algorithm PPH-CC takes as input an $n \times m$ Genotype matrix M, and a single component C of $G(J, E_f)$, and outputs an indicator function $\mathcal{E}$ for the edges in C.

j_x, j_y, j_z that are pairwise companions such that $\mathcal{E}(j_x, j_y)$ and $\mathcal{E}(j_y, j_z)$ have been set but $\mathcal{E}(j_x, j_z)$ is not set. To see this, let $E' \supseteq E_f$ contain the set of edges from $E_f \cup E_n$ whose values have been set. Consider the graph $G(J, E')$. Since the edges of E_f by themselves connect all the nodes in C, and $E_f \subseteq E'$, there is a path in $G(J, E')$ between the endpoints of any edge in C. Let $(j_x, j_z) \in C$ be such that $\mathcal{E}(j_x, j_z)$ is not set, and the path in $G(J, E')$ between its endpoints has as few edges as any path in $G(J, E')$ between the endpoints of an edge in C whose value has not yet been set. We claim that the path between j_x and j_z in $G(J, E')$ is of length two. If it is larger than two, then let j and j' respectively denote the neighbors of j_x and j_z on that path. By Corollary 9, either (j, j_z) or (j', j_x) is an edge in C. If the $\mathcal{E}$ value of that edge has been set, then the path in $G(J, E')$ between j_x and j_z is of length two as claimed. If it is not set, then that edge is in C and has a shorter path in $G(J, E')$ between its endpoints than does (j_x, j_z), a contradiction. Hence, $\mathcal{E}(j_x, j_z)$ can be set, and the algorithm does set the indicator function for each edge in C.

Let $\mathcal{E}$ be the indicator function set by algorithm *PPH-CC*. We next show that, $\forall (j_x, j_y) \in E_f \cup E_n$, $\mathcal{E}_{M'}(j_x, j_y) = \mathcal{E}(j_x, j_y)$.

The proof is by induction on the number of $\mathcal{E}$ values set by the algorithm. Clearly, for each $(j_x, j_z) \in E_f$, we have $\mathcal{E}(j_x, j_z) = \mathcal{E}_{M'}(j_x, j_z)$. Inductively, assume that for every $(j_1, j_2) \in E_n$ already set by the algorithm, $\mathcal{E}(j_1, j_2) = \mathcal{E}_{M'}(j_1, j_2)$. Consider edge (j_x, j_z) where $\mathcal{E}(j_x, j_z)$ is to be set. Let column j_y be picked by the algorithm such that both $\mathcal{E}(j_x, j_y)$ and $\mathcal{E}(j_y, j_z)$ are already set; using the induction hypothesis, $\mathcal{E}(j_x, j_y) = \mathcal{E}_{M'}(j_x, j_y)$ and $\mathcal{E}(j_y, j_z) = \mathcal{E}_{M'}(j_y, j_z)$. Now, Lemma 7 establishes there exists a row i with $M[i, j_x] = M[i, j_y] = M[i, j_z] = 2$. Hence, since the indicator function $\mathcal{E}_{M'}$ is defined by a PPH solution, Lemma 2 can be applied, establishing that $\mathcal{E}_{M'}(j_x, j_z) = \mathcal{E}_{M'}(j_x, j_y) \oplus \mathcal{E}_{M'}(j_y, j_z)$; thus, $\mathcal{E}(j_x, j_z) = \mathcal{E}(j_x, j_y) \oplus \mathcal{E}(j_y, j_z) = \mathcal{E}_{M'}(j_x, j_y) \oplus \mathcal{E}_{M'}(j_y, j_z) = \mathcal{E}_{M'}(j_x, j_z)$. ♣

Corollary 11: In any single connected component C of $G(J, E_f)$ the $\mathcal{E}$ values of the edges in C are invariant over all PPH solutions, and these values are completely determined by the $\mathcal{E}$ values on the edges of E_f in C. This also shows that the non-determinism in algorithm PPH-CC is of no consequence.

Corollary 12: If each connected component of $G_M(J, E_f \cup E_n)$ is a connected component of $G(J, E_f)$, then the solution is unique if there is one. Moreover, that solution will be found by applying algorithm *PPH-CC* to each connected component.

We can now state a very useful special case where the solution is unique, if one exists. In practical haplotyping problems, it is valuable to have unique solutions, so easy identification of uniqueness is important.

Theorem 13: If every column of M contains at least one 0 and one 1 entry, then there is either no PPH solution for M, or there is exactly one solution.

Proof: Assume that every pair of columns is a pph-pair (if not then M has no PPH solution). We prove that every connected component of $G_M(J, E_f \cup E_n)$ is a connected component of $G(J, E_f)$. We show that by showing that in G, every companion pair of columns where each has a 0 and a 1 are either forced equated or forced negated.

Consider a companion pair of columns j, j'. If all 0's in j are opposite 0's in j' and all 1's in j are opposite 1's in j', then j and j' are forced equated. So w.l.o.g assume that some 0 in j is opposite either a 1 or a 2 in j' (the other case is symmetric). Therefore, there will be a 01 row in every expansion of j, j'. Now consider the 0's in j'. If any 0 in j' is opposite a 1 or a 2 in j, then every expansion of j, j' will have a 10 row and a 01 row, so j, j' will be forced negated. Hence every 0 in j' is opposite a 0 in j. So, we now know that there is a 01 row and a 00 row in every expansion of j, j'. Now consider a 1 entry in column j. If it is opposite a 1 in j', then j and j' are forced equated. If it is opposite a 0 in j', then j and j' are forced negated, and if it is opposite a 2, then there is a complete-pair submatrix in j, j', a contradiction. The theorem now follows by applying Corollary 12. ♣

The rooted version of Theorem 13 was established in [9]. Using methods discussed there, the unique PPH solution can be found in $O(nm)$ time, when either version of this theorem applies.

4 Algorithm for PPH: setting indicator values across connected components

In the previous section, we showed that the $\mathcal{E}$ value for any edge inside a connected component of $G(J, E_f)$ is invariant over all PPH solutions. Hence the only variability in the PPH solutions are in the $\mathcal{E}$ values of edges that cross components of $G(J, E_f)$. It follows also that if there is more than one connected component of $G_M(J, E_f \cup E_n)$, then each of those components can be solved independently of the others. So, assume from this point that $G_M(J, E_f \cup E_n)$ has only one connected component and that component contains more than one component of $G(J, E_f)$.

Conceptually, shrink each component of $G(J, E_f)$ to a single node, creating a (multi-)graph denoted H. Note that each edge in H is in E_n. Choose an arbitrary spanning tree T of H, and let E_T be the edges in T. Note that if an edge e is one of several parallel edges in H, and e is in E_T, then none of the other edges parallel to e are in E_T.

Theorem 14: Consider a genotype matrix M which has at least one PPH solution. Let M' be an any of the PPH solutions for M, and $\mathcal{E}_{\mathcal{M}'}$ be the indicator function defined by M'. Then the $\mathcal{E}_{\mathcal{M}'}$ values for the edges in E_T determine the $\mathcal{E}_{\mathcal{M}'}$ values for all the edges of H in $G_M(J, E_f \cup E_n)$.

Proof: First, consider an edge (j, j') of E_T and a row i in M such that j and j' are companion columns for i. Row i expands to two rows in M', each only containing 0's and 1's. If we append those two binary rows to M, and to M', then the new M' is a PPH solution to the new M. Further, every pair that was previously in E_f remains in the new E_f with the same forced $\mathcal{E}$ value. However, the pair (j, j') which had previously been in E_n is now in E_f, and the forced $\mathcal{E}(j, j')$ value is determined by whether j and j' are equated or negated in M'. The key point is that now all the nodes in G are in a single connected component of the new $G(J, E_f)$, and so the conditions of Corollary 11 apply, and the Theorem follows from that application. ♣

Corollary 15: Let $r < m$ be the number of edges in E_T. Then the number of solutions to the PPH problem is at most 2^r.

Proof: Each solution M' determines exactly one indicator function, and those indicator values are invariant except for edges of H. Since the setting of the r edges in E_T determine the other settings, there can be at most 2^r indicator functions determined by PPH solutions, and hence at most 2^r PPH solutions. ♣

Theorem 16: For an instance M, either there are no solutions to the PPH problem for M, or any arbitrary setting of the $\mathcal{E}$ values for the edges in E_T defines a PPH solution. Hence there are exactly 2^r PPH solutions, if there are any.

Proof: Assume M has a PPH solution. Let M' be an any PPH solution for M, and $\mathcal{E}_{\mathcal{M}'}$ be the indicator function defined by M'. Let e be any edge in E_T, and let E_e be the subset of edges of H that cross the cut created by removing edge e from tree T. Note that e is in E_e and that all edges in E_e are in H.

Now modify $\mathcal{E}_{\mathcal{M}'}$ by reversing the setting of every edge in E_e. We claim that this gives values for the indicator function which satisfy Theorem 5, and hence define another PPH solution. First, since all edges in E_e are in H, the setting for every edge in E_f is unchanged. Next consider a triangle j_x, j_y, j_z in G such that $M[i, j_x] = M[i, j_y] = M[i, j_z] = 2$ for some i. If none of those edges are in E_e, then the indicator setting for them is unchanged, and the second condition of Theorem 5 is satisfied, since it was in $\mathcal{E}_{\mathcal{M}'}$. Otherwise, the triangle must contain exactly two edges from E_e, since those edges form a cycle that crosses a cut. Focus on the edge e' in the triangle that is not in E_e. If e' is set to 1 in $\mathcal{E}_{\mathcal{M}'}$, then exactly one of the other two edges is set to 1 and the other to 0. If e' is set to 0 in $\mathcal{E}_{\mathcal{M}'}$, then either both of the other edges are set to 0, or both are set to 1. In all of these cases, the second condition of Theorem 5 is satisfied after the change in indicator function is made.

Hence there is a PPH solution corresponding to the new setting of the indicator function. This proves the theorem, since any arbitrary setting of the indicator function for the edges in E_T can be created by successive edge changes, and by the above argument, each such change leads to a PPH solution. ♣

Given Theorem 16, the best we can conclude in terms of m is that the PPH problem has at most 2^{m-1} solutions. We now prove a bound that frequently will be smaller for real data. This bound is of interest both because it reduces the number of solutions, and because it can be trivially computed in practice by hand.

Theorem 17: If M has k columns that each contain both a 0 and a 1 (and a 2 by prior assumption), then M has at most $2^{(m-k-1)}$ PPH solutions.

Proof: By the proof of Theorem 13, the columns of M which each contain both a 0 and a 1, are partitioned into disconnected components of $G_M(J, E_f \cup E_n)$, and each is a connected component of $G(J, E_f)$. Hence, there are no edges of E_T which touch any node representing such a column. Therefore, the r edges in E_T form a spanning tree that can contain at most $m - k$ nodes, and hence $r \leq m - k - 1$. ♣

For the rooted PPH problem with input M and V, this theorem specializes to the following: if k' is the number of columns containing an entry which is opposite to the ancestral state for that column, then the number of PPH solutions rooted at V is at most $2^{m-k'-1}$.

Theorem 18: Algorithm PPH correctly finds a PPH solution, if there is one.

Proof: From Corollary 12, it follows that the algorithm sets the indicator function correctly on the edges in each component of $G(J, E_f)$. Assume w.l.o.g. that $G_M(J, E_n \cup E_f)$ has a single component. Now, the process of setting the indicator function value on the edges in E_T to 0 can be

```
Algorithm PPH:
Input: An $n \times m$ genotype matrix M
Output: 0-1 valued functions $\mathcal{E}$ defined on each companion pair of columns $j_1, j_2$.
Algorithm:
    Set:
        1. Using algorithm PPH-CC, set each edge in $E_n$ that is within a connected component
           of $G(J, E_f)$.
        2. Select a set of edges $E_T$ for the spanning tree $T$ of $H$, and set the $\mathcal{E}$ value for each one
           to zero.
        3. For each triple column $j_1, j_2, j_3$ such that $\mathcal{E}(j_1, j_2)$, and $\mathcal{E}(j_2, j_3)$ are set, but $\mathcal{E}(j_1, j_3)$
           is not and $j_1, j_3$ are companions, set $\mathcal{E}(j_1, j_3) = \mathcal{E}(j_1, j_2) \oplus \mathcal{E}(j_1, j_2)$.
    Output: Output $\mathcal{E}$
```

Figure 5: Algorithm PPH takes as input an $n \times m$ Genotype matrix M and outputs an indicator function $\mathcal{E}$.

thought of as moving these edges from E_n to E_f. The resulting graph G' has a single component when restricted to the new E_f set. From Corollary 12, if there is a solution for G', the algorithm correctly finds it. Also, from Theorem 16, it follows that if a solution does not exist for this setting of the edges from E_T, then no other setting of the indicator function on the edges from E_T works either, and thus no solution exists. ♣

5 Efficient Implementation of Algorithm *PPH*

In this section, we discuss implementation issues with respect to setting the edges and validating the output haplotype matrix M'. First, note that validating that an $n \times m$ binary matrix M' satisfies a perfect phylogeny can be done $O(nm)$ time [6, 7]. The critical components of Algorithm *MAIN_PPH* are the procedures *BuildGraph* and *SetEdges*, which are described below. The pseudo-code description is given in Figure 6 and Figure 7.

The procedure *BuildGraph* (see Figure 6) constructs the genotype graph $G_M(J, E_f \cup E_n)$ from the input genotype matrix M. Every pair of columns is checked to see if it is a *pph*-pair. Note that if there is a pair that is not a *pph*-pair, then no solution exists. For companion pairs j', j, the edge $(j', j) \in E_f$ if the rows that are not companion rows for j', j, expand to produce either $\{00, 11\}$ or $\{01, 10\}$; for companion pairs j', j, $(j', j) \in E_n$ otherwise. If the edge is in E_f, then its $\mathcal{E}$ value is set 1 or 0 appropriately. *BuildGraph* runs in $O(nm^2)$.

The procedure *SetEdges* sets the $\mathcal{E}$ values for the edges in E_n. It operates on each connected component of G_M separately. Each connected component contains components of $G = (J, E_f)$ as subgraphs and thus can be handled using the observations made in Section 4. *SetEdges* is a Depth-First-Search (DFS) like procedure that traverses G_M using only the edges in E_f. Since the traversal is done using only edges in E_f, unlike traditional DFS trees, our DFS tree can have both back edges and cross edges; a back edge is in $E_f \cup E_n$ whereas a cross edge is in E_n.

Let C denote the set of nodes visited so far in the current connected component of G_M being processed and P denote the current path in the DFS tree for this component. We use set E_T to denote the set of edges that putatively belong to the spanning tree T of H (where H is defined as in Section 4). For a node v, we use $prev[v]$ to denote the parent of v in the DFS tree and $next[v]$ to denote the child of v in the current path P of the DFS tree. In our DFS traversal procedure, at the end of visiting a node v, we will have set the $\mathcal{E}$ value of every edge $(v, u) \in E_n$ where $u \in C$.

Suppose we are at a node v in the DFS tree. Then we add v to both C and P. We first process all $(v,u) \in E_n$ before processing $(v,u) \in E_f$. For each neighbor u of v, we check to see if $(v,u) \in E_n$ is a back edge (i.e $u \in P$) or a cross edge (i.e. $u \notin P$ but $u \in C$). If u has not been seen before (i.e. $u \notin C$), then (u,v) is neither and is added to E_T; thus E_T contains edges such that one node of the edge is not currently in the component C.

Consider the case when $(v,u) \in E_n$ is a back edge and $\mathcal{E}(v,u)$ has not been set. Note that there is a cycle in G_M containing the edge $(v,u) \in E_n$. From Corollary 9, either $(prev[v],u) \in E_f \cup E_n$ or $(v, next[u]) \in E_f \cup E_n$. If it is the former, then $\mathcal{E}(prev[v],u)$ is already set as we visited $prev[v]$ before v. Thus, $\mathcal{E}(v,u) = \mathcal{E}(\text{prev}[v],u) \oplus \mathcal{E}(v,\text{prev}[v])$. If it is the later and $\mathcal{E}(v,next[u])$ has not been set yet, then we push u onto a stack S, and set $u = next[u]$. In this way, we traverse the path P starting at u, till we reach a u for which either $\mathcal{E}(v,u)$ is set or $(\text{prev}[v],u) \in E_f \cup E_n$. Once this happens, then S is repeatedly popped (say u denotes the element that is popped) and $\mathcal{E}(v,u)$ is set to $\mathcal{E}(v,\text{next}[u]) \oplus \mathcal{E}(u,\text{next}[u])$. Also, every time the $\mathcal{E}$ value is set for an edge, this edge is removed from E_T (if it is present in E_T).

The case when $(v,u) \in E_n$ is a cross edge is also handled in a similar fashion using the observation of Corollary 9 and a similar stack data structure. In this manner, every edge from E_n that goes between two nodes of the same component in $G = (J, E_f)$ can handled. Once such a component is processed, then we pick some edge (v,u) from E_T, and set $\mathcal{E}(v,u) = 0$. Note that u and v belong to different components in $G = (J, E_f)$. Suppose we have just processed the component containing v. In order for the DFS algorithm to process the component of $G = (J, E_f)$ containing u, we assign (v,u) to be in E_f, and proceed to consider the back edges and cross edges as above.

The complete pseudo-code description is given in Figure 7. Note that every node and edge in G_M is visited at most a constant number of times. Furthermore, it is possible to implement the push, pop, and remove operations for E_T in constant time for each edge using a combination of an adjacency matrix and a doubly linked list. Thus the total time for *SetEdges* is $O(m + |E_f \cup E_n|)$; this is $O(m^2)$. Consequently, the running time of Algorithm PPH' is bounded by $O(nm^2)$.

6 Conclusions and Future Work

We have described an algorithm for SNP haplotyping that is simple to implement and efficient. As building the graph is now the computational bottleneck, the obvious open technical problem is whether it can be built in $o(nm^2)$ time. We conjecture that $O(nm)$ is achievable for this algorithm.

There are many HI algorithms in the literature (See [2, 13] for examples) which work well in practice, but give no guarantees about the resulting haplotype matrix. In practice, people use one or more of these algorithms and then test to see if the resulting haplotype matrix forms a perfect phylogeny. Our solution attempts to turn this around by first testing to see if a prefect phylogeny solution is possible. If none is, we can resort to inferring haplotypes using other methods.

References

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

[2] Andrew Clark. Inference of haplotypes from PCR-amplified samples of diploid populations. *Mol. Biol. Evol*, 7:111–122, 1990.

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

```
**Main** Algorithm $MAIN\_PPH$(M)
    $G_M(J, E_f \cup E_n)$= BuildGraph(M)
    Set $E_T = \phi$                    #*Putative Edges from spanning tree T*
    Set $C = \phi$                      #*Current Connected Component*
    Set $P = \phi$                      #*Current DFS Path*
    for all $v \in J$ search[v]=FALSE
    for all $(u, v) \in E_n$ set $\mathcal{E}(u, v) = *$
    for all $v \in J$
        if (search[v]=FALSE)                        #*Step (i)*
            SetEdges(v)
            while ($E_T$.notEmpty())
                (u,v) = $E_T$.pop()
                $\mathcal{E}(u, v) = 0$
                Add $(u, v)$ to $E_f$; Remove $(u, v)$ from $E_n$
                if $u \in C$, SetEdges(v)        else SetEdges(u)
            end while
        end if
    end for
    Set M' = $\mathcal{E}2M(M, \mathcal{E})$
    if Valid(M') return M' else return error $\phi$
Procedure BuildGraph(M)
    Set $J = \phi$; Set $E_f = \phi$; Set $E_n = \phi$
    for $j = 1 \ldots m$
        $J = J + \{j\}$
        for $j' = 1 \ldots j - 1$
            for $i = 1 \ldots n$
                if $M[i, (j', j)] = 22$, Set Is22(j',j)=TRUE
                else
                    if $M[i, (j', j)]$ expands to 00, Set Is00(j',j)=TRUE
                    if $M[i, (j', j)]$ expands to 01, Set Is01(j',j)=TRUE
                    if $M[i, (j', j)]$ expands to 10, Set Is10(j',j)=TRUE
                    if $M[i, (j', j)]$ expands to 11, Set Is11(j',j)=TRUE
                end if
            end for
            if (Is00(j,j') && Is01(j,j') && Is10(j,j') && Is11(j,j'))
                Return $\phi$ and exit                    #*Complete Pair Submatrix*
            else if(Is00(j,j') && Is11(j,j') && Is22(j,j'))
                $E_f = E_f + \{(j', j)\}$
                $\mathcal{E}(j', j) = 0$
            else if(Is01(j,j') && Is10(j,j') && Is22(j,j'))
                $E_f = E_f + \{(j', j)\}$              $\mathcal{E}(j', j) = 1$
            else if(Is22(j,j'))
                $E_n = E_n + \{(j', j)\}$
            end if
        end for
    end for
```

Figure 6: An $O(nm^2)$ implementation of Algorithm *PPH*

```
Procedure SetEdges(vertex v)
P.add(v);  C.add(v)
search[v]=TRUE
for all $u$ s.t. $(v,u) \in E_n$ && $(\mathcal{E}(u,v) == *)$
    if (( $u \in P$))                          #Back Edge
        Stack $S = \phi$
        while $(\mathcal{E}(v,u) == *)$ && $((\text{prev}[v], u) \notin E_f \cup E_n)$
            S.push(u)
            u=next[u]
        end while
        if $(\mathcal{E}(v,u) == *)$
            $\mathcal{E}(v,u) = \mathcal{E}(\text{prev}[v], u) \oplus \mathcal{E}(v, \text{prev}[v])$
        while(S.notEmpty())
            u = S.pop()
            $\mathcal{E}(v,u) = \mathcal{E}(v, \text{next}[u]) \oplus \mathcal{E}(u, \text{next}[u])$
            $E_T$.remove($(v,u)$)
        end while
    else if (($u \in C$))                          #Cross Edge
        Stack $S = \phi$
        while $(\mathcal{E}(v,u) == *)$ && $((\text{prev}[v], u) \notin E_f \cup E_n)$
            S.push(u)
            u=prev[u]
        end while
        if $(\mathcal{E}(v,u) == *)$
            $\mathcal{E}(v,u) = \mathcal{E}(\text{prev}[v], u) \oplus \mathcal{E}(v, \text{prev}[v])$
        while (S.notEmpty())
            u = S.pop()
            $\mathcal{E}(v,u) = \mathcal{E}(v, \text{next}[u]) \oplus \mathcal{E}(u, \text{next}[u])$
            $E_T$.remove($(v,u)$)
        end while
    else                          #Putative edge from $E_T$
        $E_T$.push($(v,u)$)
    end if
end for
for all $u$ s.t. $(v,u) \in E_f$ && (search[u] == FALSE)                          #Tree Edge
    next[v]=u;  prev[u]=v
    SetEdges(u)
end for
P.remove(v)
```

Figure 7: An $O(m^2)$ implementation of Procedure SetEdges

[4] L. Friss, R. Hudson, A. Bartoszewicz, J. Wall, T. Donfalk, and A. Di Rienzo. Gene conversion and differential population histories may explain the contrast between polymorphism and linkage disequilibrium levels. *Am. J. of Human Genetics*, 69:831–843, 2001.

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

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

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

[8] D. Gusfield. Inference of haplotypes from samples of diploid populations: complexity and algorithms. *Journal of computational biology*, 8(3), 2001.

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

[10] L. Helmuth. Genome research: Map of the human genome 3.0. *Science*, 293(5530):583–585, 2001.

[11] S. Orzack, D. Gusfield, and V. Stanton. The absolute and relative accuracy of haplotype inferral methods and a consensus approach to haplotype inferral, 2001. Abstract Nr 115 in Am. Society of Human Genetics, Supplement 2001.

[12] J. C. Stephens and et al. Haplotype variation and linkage disequilibrium in 313 human genes. *Science*, 293:489–493, 2001.

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

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