

Haplotyping as Perfect Phylogeny: A direct approach

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

February 7, 2003

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 Haplotype Map project has been announced by NIH. The biological key to that project is the surprising fact that some Human 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 [16, 4, 21, 9].

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, motivates 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 [14] - given n genotypes of length m each, does there exist a set of at most $2n$ haplotypes such that each

*The Center for Advancement of Genomics, 1901 Research Blvd., 6th floor, Rockville, MD 20850. Ph: 301-309-3421. Fax: 301-309-3434. Email: vbafna@tcag.org

†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 and EIA-0220154

‡DEI, University of Padova, Via Gradenigo, 6-A, 35131 Padova, Italy. Email: lancia@dei.unipd.it

§Informatics Research, Applied Biosystems, 45 W. Gude Drive, Rockville, MD 20850. Email:

Shibu.Yooseph@celera.com

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 [14] to solve this problem reduces it to established, deep, results and algorithms from matroid and graph theory. Although that reduction is quite simple, and the resulting algorithm nearly optimal in speed, 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. However, as stated in [14], many simplifications are possible and the list of “future work” in [14] began with the task of developing a simpler more direct, yet still efficient algorithm. This paper accomplishes that goal, for both the rooted and unrooted PPH problems. It establishes a 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, self-contained proofs. In addition to algorithmic simplicity, the approach here makes the representation of all solutions more intuitive than in [14], and solves another 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 [16]. It will be used in large-scale screens of populations to associate specific haplotypes with specific complex genetic-influenced diseases. A Haplotype Map strategy is presently being finalized by an NIH working-group. The biological key to that strategy is the surprising fact that some Human 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 [16, 4, 21, 9].

Mathematically, this lack of recombination (along with the infinite sites assumption) motivates 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 [23, 14]. 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 of length m each, 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 [14] showed that the PPH problem can be solved in $O(nm\alpha(nm))$ time, where α is the inverse Ackerman function, and hence this time bound is almost linear in the input size, nm . It was also shown in [14], 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, 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. However, the "Future Work" section of [14] stated: "It is clear that many simplifications can be made for the special case of the PPH

problem, and future work will find such simplifications ...”.

Here we develop a simpler more direct, yet still efficient, algorithm, via a self-contained approach (not needing deep matroid or graph theorems). This is achieved for both the rooted and unrooted PPH problems. We establish a 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 [14], and solves another 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.

Most of the material in this paper originates in [1]. Independently, Eskin, Halperin and Karp [5], developed an $O(nm^2)$ -time algorithm for the PPH problem, and established that time bound in [7]. They also extended their algorithm to handle parent-child genotype data [5], and extended their PPH program to handle genotype data that does not fit the perfect phylogeny model, but fits the model if small modifications in the data are made [6].

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 [3, 2, 10, 22, 13, 20, 19, 8, 15, 18, 17]. Some of these methods are extremely accurate (in the 80-95%) range, but improvements in accuracy, speed and range of applicability are still desired.

1.3 The Haplotype Inference (HI) problem

Given n genotype vectors, the *Haplotype Inference (HI)* problem is to find n pairs of haplotype vectors that could have generated the genotype vectors. 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 $\prod_{i=1}^n 2^{h_i-1}$ 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 a $2n \times m$ 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 motivation for the perfect phylogeny model is based on recent observations of no or little recombination in long segments of Human DNA, and the standard infinite-sites assumption. See [23, 14] 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 [14]. 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 [14] 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.

1.4 A road map

In the following, we use different but equivalent representations of the problem and switch between them to expose our solution to the PPH problem. However, the key ideas are straightforward. Note first that for every row i , and columns j_1, j_2 of M such that $M[i, j_1] = M[i, j_2] = 2$, we have exactly two choices of resolving into haplotypes. The resolution either leads to the submatrix containing 00 and 11, or 01 and 10. In section 2, we establish that for every pair of columns, the resolution must be identical for all rows. Therefore, instead of resolving each row independently, we can talk about resolving pairs of columns.

In section 3, we construct an auxiliary graph whose nodes correspond to the columns of the matrix, and edges correspond to pairs of columns which need to be resolved. Edges are labeled 0 or 1 depending on the choice of resolution. Some of these edge labels are *forced*, and others are *inferred* based on the forced edges. For the remaining edges which are neither forced nor inferred, we show in section 4 that we can choose a subset such that either no choice of labels allows a perfect phylogeny, or every choice of labels gives a distinct solution to the PPH problem. This gives us a representation of all possible solutions, a simple mechanism for switching between solutions, and also gives a computable formula for the number of solutions. Finally, in section 5, we show how to implement the edge labeling efficiently.

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 [11, 12] 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 .

	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.

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 [3, 23]. This is called the “four-gamete test” in the genetics 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 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 $J = \{j_1, j_2, \dots, j_l\}$ of a genotype matrix M , with a companion row i (i.e. $M[i, j_1] = M[i, j_2] = \dots = M[i, j_l] = 2$). 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: Let M' be a solution to the PPH problem for a genotype matrix M . 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. In more biological terms, two columns are equated if they are “in phase”, and are negated when they are “out of phase”. 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. $\mathcal{E}(j_x, j_z) = \mathcal{E}(j_x, j_y) \oplus \mathcal{E}(j_y, j_z) \quad \forall j_x, j_y, j_z$ 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

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

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

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, let k be any index less than j such that $M[i, k] = 2$; 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'

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.

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

We use the characterization in the previous section to define an algorithm for the PPH problem. Therefore, we will describe the algorithm mainly in terms of setting the indicator functions \mathcal{E} .

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 $J = \{j_1, \dots, j_n\}$ 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' 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. $M[i_2, j_3] = M[i_2, j_2] = 2 \neq M[i_2, j_1]$
3. $M[i_3, j_1] = M[i_3, j_3] = 2 \neq M[i_3, j_2]$

Thus, the submatrix $M[*, (j_1, j_3)]$ contains the forcing pattern $\{x2, 2y\}$, where $x, y \in \{0, 1\}$. By Lemma 4, $(j_1, j_3) \in E_f$, 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. ♣

Theorem 8 can be seen to hold for any path of length 3 or more, where edge (j, j') is not at either end of the path. However, we will not use that fact in this paper.

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, \dots, 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 $(j_2, j_k) \in E_f \cup E_n$.

	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 $(j, j_2) \in E_f \cup E_n$.

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 $\mathcal{E}_{M'}$ 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 $\mathcal{E}_{M'}$.

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 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 .

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 $\mathcal{E}(j_1, j_2) \in C$, 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 $\mathcal{E}(j_1, j_3) = \mathcal{E}(j_1, j_2) \oplus \mathcal{E}(j_2, j_3)$

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 .

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 [14]. 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 any of the PPH solutions for M , and $\mathcal{E}_{M'}$ be the indicator function defined by M' . Then the $\mathcal{E}_{M'}$ values for the edges in E_T determine the $\mathcal{E}_{M'}$ 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 any PPH solution for M , and $\mathcal{E}_{M'}$ 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, we have an exact formula for the number of solutions to the PPH problem. However, this formula can only be computed by solving the PPH problem explicitly. Also, 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 will frequently be smaller for real data. This bound is of interest both because it often reduces the number of solutions from 2^{m-1} , and because it can be trivially computed in practice by hand. We first need the following lemma.

Lemma 17: In any connected component of $G_M(E_f \cup E_n)$, all the columns that contain both a 0 and a 1 are in the same connected component of E_f .

Proof: In Theorem 13 we showed that any companion pair of columns that each contain a 0 and

x	j_1	j_2	\dots	j_l	y
2	2	-	\dots	-	-
-	2	2	\dots	-	-
-	-	2	\dots	-	-
-	-	-	\dots	2	2

Figure 5: A submatrix describing a shortest path from x to y in $G_M(J, E_f \cup E_n)$.

a 1 are either forced negated or forced equated, and are thus in the same component of E_f . Now consider two columns x and y that each contain a 0 and a 1, that are *not* companion columns, but are in the same component of $G_M(E_f \cup E_n)$. There must be a path connecting x and y , where each adjacent pair of columns is a companion pair. Let $x, j_1, j_2, \dots, j_l, y$ be a shortest such path between x and y . The matrix looks like the one given in Figure 5, where none of the cells marked with - has a 2 (otherwise a shorter path exists), and the matrix has at least three rows (as x contains a 0, 1 and a 2). We will show that each edge on this path is in E_f .

By Lemma 4, each of the consecutive pairs j_i, j_{i+1} in the path contains a forcing-pattern, and hence each such associated edge is in E_f . So, we only need to show that the two end edges are in E_f . The two cases are symmetric, so we focus on (x, j_1) .

First, suppose the entry in the second row of column x is a 0, so there is at least one row below row two where the value in column x is 1. If the value in that row in column j_1 is 1, then x and j_1 are forced equated, and if the value is 0, then they are forced negated. Symmetrically, if the entry in the second row of x is a 1, then there is a 0 in column x below the second row. If the value in that row of j_1 is 1, then x and j_1 are forced negated, and if that value is 0, then they are forced equated. Since, the value cannot be 2, in either case edge (x, j_1) is in E_f .

♣

Theorem 18: If there are k columns of M which each contain both a 0 and a 1, then the number

of PPH solutions for M is at most 2^{m-k} .

Proof:

By Lemma 17, in any connected component of $G_M(E_f \cup E_n)$, all the columns that contain both a 0 and 1 are connected together in a connected component of E_f . Suppose there are k_i such columns in component i of $G_M(E_f \cup E_n)$, and there are m_i columns overall in component i , and there are r_i edges of E_T in that component. Then r_i is bounded by $m_i - k_i + 1 - 1 = m_i - k_i$. (The +1 is for the component of E_f that includes the columns with both 0 and 1, the -1 is because edges in E_T form a tree). Over all components of $G_M(E_f \cup E_n)$, r is bounded by the $\sum r_i$, which is bounded by $\sum(m_i - k_i) = m - k$. By Theorem 16, the number of solutions is either 0 or exactly 2^r where r is the number of edges in E_T , so the bound of 2^{m-k} follows. Note that 2^{m-1} is a general bound, which is lower when $k = 0$. ♣

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 $2^{m-k'}$.

Theorem 19: 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 $G_M(J, E_f \cup E_n)$ has a single component. Now, the process of setting the indicator function value on the edges in E_T to 0 can be 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. ♣

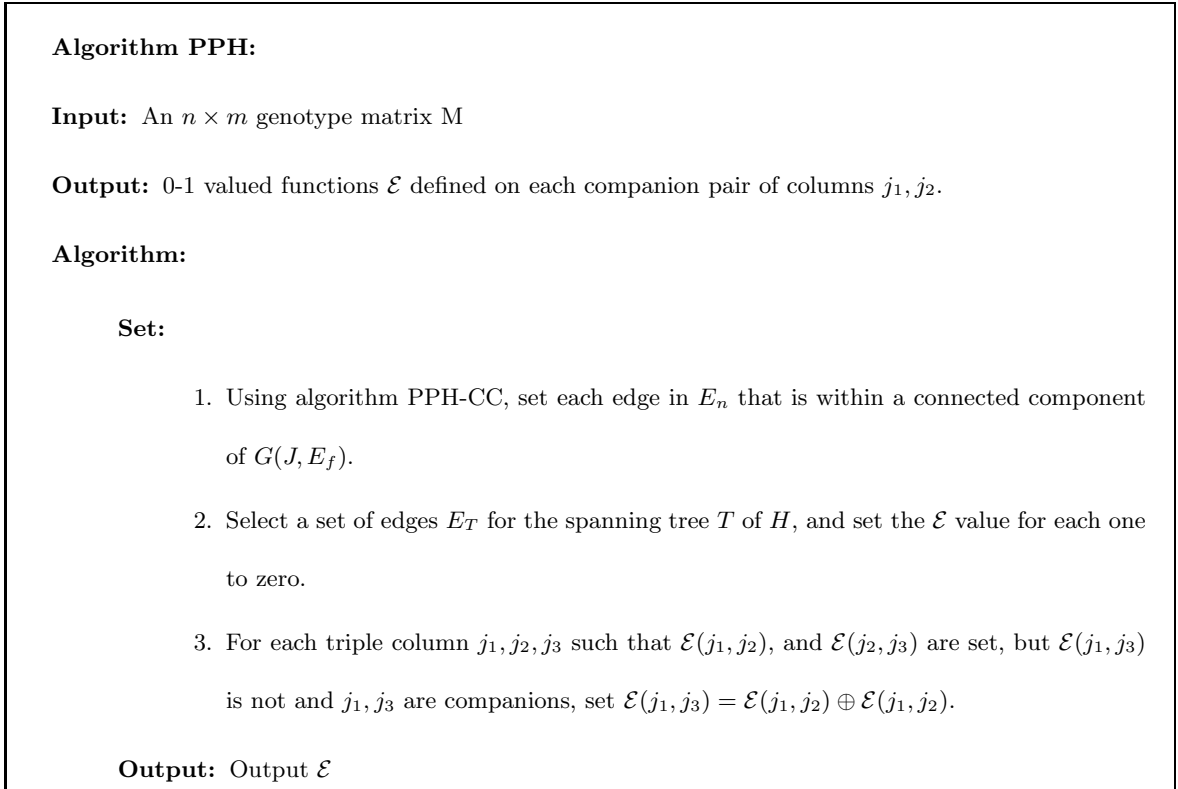


Figure 6: Algorithm PPH takes as input an $n \times m$ Genotype matrix M and outputs an indicator function \mathcal{E} .

5 An efficient algorithm for the PPH problem

In this section we describe an $O(nm^2)$ algorithm that outputs a haplotype matrix M' that is a PPH solution (if one exists) for the given $n \times m$ input genotype matrix M . The algorithm has four main steps:

1. Construct the graph $G_M = (J, E_f \cup E_n)$ from the genotype matrix M and set the indicator function \mathcal{E} appropriately for each edge in E_f .
2. Set the indicator function \mathcal{E} appropriately for each edge in E_n .
3. Use \mathcal{E} to produce a $2n \times m$ haplotype matrix M' from the genotype matrix M .
4. Check M' to see if it admits a perfect phylogeny.

Step 1 is accomplished by the procedure *BuildGraph* (see Figure 7). Recall that J , the node set of G_M , consists of the set of columns in M . Every pair of columns is checked to see if it is a *pph*-pair. If there is a pair that is not a *pph*-pair, then no solution exists. Hence, the algorithm proceeds on the assumption that all companion pairs are *pph*-pairs. For companion pairs j', j , the edge (j', j) is placed 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) is placed 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 $O(nm^2)$.

Step 3 is accomplished by the algorithm $\mathcal{E}2M$ described in Figure 2; it runs in $O(nm)$. Step 4 can be implemented in $O(nm)$ using the perfect phylogeny recognition algorithm for binary characters [11, 12].

We now show that Step 2 can be implemented in $O(m^2)$. This implies a total running time of $O(nm^2)$ for the entire algorithm.

In Step 2, we operate on each connected component of $G_M = (J, E_f \cup E_n)$ separately. Let C denote a connected component of G_M and let ST denote a spanning tree of C . Identify the connected components of C induced by its edges that belong to E_f ; let S_i denote such a component

of C . Identify the edge set $E_C = \{(j_1, j_2) | (j_1, j_2) \text{ is an edge in the spanning tree } ST \text{ and such that } j_1 \in S_{i_1}, j_2 \in S_{i_2}, \text{ and } i_1 \neq i_2\}$. By definition, each edge in E_C is in E_n . The set E_C can be identified in time linear in the size of C by performing simple graph traversals (either breadth-first-search or depth-first-search). Note that E_C is the same as the edge set E_T that is used in Theorem 14 and Theorem 16; we have chosen to define it slightly differently in the implementation. From Theorem 14 it follows that setting the \mathcal{E} values on the edges in E_C forces the \mathcal{E} values of the rest of the edges of C that belong to E_n . Thus we proceed to move the edges in E_C from E_n to E_f (by setting each edge to a 0). As a result, the edges of component C that belong to E_f induce a single component. Procedure *SetEdges* (see Figure 8) shows how to set all the edges of this C that belong to E_n , in time linear in the size of C .

SetEdges is a Depth-First-Search (DFS) procedure that traverses C using only its edges that belong to 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 . In our DFS 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 has already been visited. We use P to denote the current path in the DFS tree. 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 DFS tree.

Suppose we are at a node v in the DFS tree. 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 has been visited already).

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 component C 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, $\mathcal{E}(v, u) = \mathcal{E}(prev[v], u) \oplus \mathcal{E}(v, prev[v])$. 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 $(prev[v], u) \in E_f \cup E_n$. Once

this happens, then S is repeatedly popped (say u denotes the element that is popped) and $\mathcal{E}(v, u)$ is set to $\mathcal{E}(v, \text{next}[u]) \oplus \mathcal{E}(u, \text{next}[u])$.

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. The complete pseudo-code description is given in Figure 8. The total running time on all the components of G_M is thus bounded by $O(|J| + |E_f \cup E_n|)$; this is $O(m^2)$.

For instances with multiple solutions, the set of all solutions can be generated as follows. Let $\mathcal{A} = \{(j, j') \mid (j, j') \text{ is an edge in } E_C, \text{ where } C \text{ is a connected component of } G_M\}$. There are $2^{|\mathcal{A}|}$ settings of \mathcal{E} on the edges in \mathcal{A} , where each setting consists of assigning $\mathcal{E}(j, j')$ a 0 or a 1 (for $(j, j') \in \mathcal{A}$). From Theorem 16, each of these settings generates a different solution. If p denotes the number of connected components of G_M and q denotes the number of connected components in $G = (J, E_f)$, then $|\mathcal{A}| = q - p$.

This algorithm has been fully implemented by R.H. Chung, and is available as program DPPH from www.cs.ucdavis.edu/~gusfield/. It solves problem instances of size 50 genotypes and 100 sites in under one second, and problem instances of size 500 genotypes and 1000 sites in under two minutes on an 800 Mhz G4 computer.

6 Secondary Optimization Problems

The representation of all solutions allows the framing and (sometimes) the efficient solution of secondary optimization questions. For example, random PPH solutions can be efficiently and uniformly generated by picking any spanning forest of H and randomly setting the indicator variables on its edges. Of most importance, the representation explicitly and completely determines whether additional phase or partial haplotype information is redundant or not. Consider the problem where the genotype matrix M comes from a set of chromosome pairs that can be explored in the laboratory, but at significant cost. In the lab, one can examine the two chromosomes that correspond to a genotype i , and for any pair of sites (j, j') such that $M[i, j] = M[i, j'] = 2$, determine whether sites

```

Main Algorithm MAIN_PPH(M)
   $G_M(J, E_f \cup E_n) = \text{BuildGraph}(M)$ 
  for all  $v \in J$   $\text{search}[v] = \text{FALSE}$ 
  for all  $(u, v) \in E_n$   $\text{set } \mathcal{E}(u, v) = *$ 
  Identify connected components of  $G_M$ 
  for all connected components  $C$  of  $G_M$ 
    Identify  $E_C$ 
    for all  $(u, v) \in E_C$ ,  $\text{set } \mathcal{E}(u, v) = 0$ 
     $\text{Set } P = \phi$  # Current DFS Path
    Pick a node  $v \in C$ 
     $\text{SetEdges}(v)$ 
  end for
   $\text{Set } M' = \mathcal{E}2M(M, \mathcal{E})$ 
  if  $\text{Check}(M')$  return  $M'$  else return error  $\phi$  # Perfect Phylogeny recognition algo

Procedure BuildGraph(M)
   $\text{Set } J = \phi$ ;  $\text{Set } E_f = \phi$ ;  $\text{Set } E_n = \phi$ 
  for  $j = 1 \dots m$ 
     $J = J + \{j\}$ 
    for  $j' = 1 \dots j - 1$ 
      for  $i = 1 \dots n$ 
        if  $M[i, (j', j)] = 22$ ,  $\text{Set } Is22(j', j) = \text{TRUE}$ 
        else
          if  $M[i, (j', j)]$  expands to 00,  $\text{Set } Is00(j', j) = \text{TRUE}$ 
          if  $M[i, (j', j)]$  expands to 01,  $\text{Set } Is01(j', j) = \text{TRUE}$ 
          if  $M[i, (j', j)]$  expands to 10,  $\text{Set } Is10(j', j) = \text{TRUE}$ 
          if  $M[i, (j', j)]$  expands to 11,  $\text{Set } Is11(j', j) = \text{TRUE}$ 
        end if
      end for
      if  $(Is00(j, j') \ \&\& \ Is01(j, j') \ \&\& \ Is10(j, j') \ \&\& \ Is11(j, j'))$ 
         $\text{Return } \phi$  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
end for

```

Figure 7: An $O(nm^2)$ implementation of Algorithm *PPH*

```

Procedure SetEdges(vertex v)
P.add(v)           #Add to current path
search[v]=TRUE
for all u s.t. (v, u) ∈ En && (E(u, v) == *)
  if (u ∈ P)           #Back Edge
    Stack S = φ
    while (E(v, u) == *) && ((prev[v], u) ∉ Ef ∪ En)
      S.push(u)
      u=next[u]
    end while
    if (E(v, u) == *)
      E(v, u) = E(prev[v], u) ⊕ E(v, prev[v])
    while(S.notEmpty())
      u = S.pop()
      E(v, u) = E(v, next[u]) ⊕ E(u, next[u])
    end while
  else if (search[u])           #Cross Edge
    Stack S = φ
    while (E(v, u) == *) && ((prev[v], u) ∉ Ef ∪ En)
      S.push(u)
      u=prev[u]
    end while
    if (E(v, u) == *)
      E(v, u) = E(prev[v], u) ⊕ E(v, prev[v])
    while (S.notEmpty())
      u = S.pop()
      E(v, u) = E(v, prev[u]) ⊕ E(u, prev[u])
    end while
  end if
end for
for all u s.t. (v, u) ∈ Ef && (search[u] == FALSE)           #DFS Tree Edge
  next[v]=u; prev[u]=v
  SetEdges(u)
end for
P.remove(v)

```

Figure 8: An $O(m^2)$ implementation of Procedure SetEdges

j and j' are set equal or set negated in the chromosomes. That is, whether these sites are “in-phase” or “out-of-phase”. This is done currently by using allele-specific PCR. A query for a triple (i, j, j') is “informative” if and only if there are two solutions to the PPH problem for M where sites j and j' are equated in one solution, and negated in the other. Clearly, the query for (i, j, j') will be informative if and only if the nodes for j and j' are endpoints of an edge in graph H . Moreover, once the answer for the query (i, j, j') is determined, the components of $G_M(J, E_f)$ containing j and j' , merge to become one component. It follows, that in order to uniquely determine the underlying perfect phylogenetic tree through such queries, the set of queries must correspond to edges creating a spanning forest of graph H . Hence the number of such queries is exactly $k - q$, where k is the number of nodes of H , and q is the number of connected components of H (and of G_M). If each query has a cost, then the minimum cost set of queries is specified by a minimum cost spanning forest.

A more difficult problem is defined by the primitive operation of specifying a row i , and completely determining in the lab, the two haplotypes that make up genotype i . The determination of the two haplotypes for a single genotype i can answer the queries for several (i, j, j') triples, and hence it is not trivial to choose the smallest set of rows. However, an optimal solution in practice may be obtained through mathematical programming. Each row determines a set of equalities that each say that if the row i is selected, then a particular triple query will be answered. Each query corresponds to an edge in H , and we need inequalities to specify that the edges corresponding to answered queries create a spanning forest of H . There are several approaches to this. For a single connected component of H , if there are k nodes, and k is modest, one can enumerate each of the 2^k subsets of nodes, and for each, create an integer programming inequality that asserts that at least one row must be selected so that at least one query crossing that cut is answered. Integer programming software such as CPLEX may be very effective in solving moderate sized problem instances that result. Alternatively, cutting plane methods that successively add inequalities to cross any uncrossed cuts may be effective. Such methods are well developed in the mathematical

programming literature. The point here is not to completely solve this mathematical programming problem, but to illustrate the utility of the representation of all solutions. The representation identifies what additional information will uniquely determine the underlying tree, allowing a finite solution to this laboratory optimization problem. Without such a representation, it is not clear which laboratory tests would be useful.

7 A more direct biologically-appealing representation

Assume for now that the graph $G_M(J, E_f \cup E_n)$ only contains a single component. Let G_f be the subgraph of G_M all nodes in J , but only containing those edges in E_f . Thus, graph G_f can have components consisting of isolated nodes. Suppose G_f contains $r > 1$ connected components. Each component C of G_f defines set of columns in M . We will refer to that set of columns by the name of its component C . So we can think of G_f as determining a fixed *partition* of the columns of M in disjoint classes. The term “class” will denote a class in this fixed partition.

Let M' be any expansion of matrix M which solves the PPH problem for M . Recall that each row $M[i, *]$ expands to two rows denoted $M'[i, *]$ and $M'[i', *]$. Relative to M' , “a class reversal” in a class C means changing the value 0 to 1, and 1 to 0 in every pair of cells $M'[i, j], M'[i', j]$, such that $M[i, j] = 2$, and $j \in C$. We claim that the result of a class reversal is another expansion of M that solves the PPH problem for M . This follows from the proof of Theorem 16. Moreover, every solution to the PPH problem can be obtained by some choice of class reversals, starting from M' .

Hence, an appealing visual representation of the set of all solutions to the PPH problem is a single solution M' , where all the columns in any class defined by a connected component of G_f are grouped together. In each class in the partition of columns, one can independently choose whether or not to make a class reversal. Each such choice leads to a solution of the PPH problem, and each solution to the PPH problem corresponds to such a choice. However, in order to enumerate each solution only once, we arbitrarily choose one class and fix the 0,1 setting in the columns of that class to be as in M' .

When graph G_M contains more than a single component, then we can work on each component independently. That is, in each component of G_M , the settings in the columns of one class (defined by a component of G_f) are held fixed, while class reversals in the other class can all be made independently of each other. Note that if a component in G_M has only a single node, its settings are never changed.

The program DPPH produces the representation discussed here.

8 Conclusions and Open Problem

We have described an algorithm for SNP haplotyping that is simple to implement and very efficient in practice. 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.

References

- [1] V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph. Haplotyping as perfect phylogeny: A direct approach. Technical report, UC Davis, Department of Computer Science, 2002.
- [2] A. Clark, K. Weiss, D. Nickerson, S. Taylor, A. Buchanan, J. Stengard, V. Salomaa, E. Var-tiainen, M. Perola, E. Boerwinkle, and C. Sing. Haplotype structure and population genetic inferences from nucleotide-sequence variation in human lipoprotein lipase. *Am. J. Human Ge-netics*, 63:595–612, 1998.
- [3] Andrew Clark. Inference of haplotypes from PCR-amplified samples of diploid populations. *Mol. Biol. Evol.*, 7:111–122, 1990.
- [4] 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.

- [5] E. Eskin, E. Halperin, and R. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. Technical report, UC Berkeley, Computer Science Division (EEDS), 2002.
- [6] E. Eskin, E. Halperin, and R. Karp. Large scale reconstruction of haplotypes from genotype data. In *Proc. 7th Int. Conf. Computational Biology*, pages 104–113, 2003.
- [7] E. Eskin, E. Halperin, and R.M. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. Technical report, Columbia University, 2002. Updated version of the UCB technical report of that title, available at <http://www.cs.columbia.edu/compbio/hap/>.
- [8] L. Excoffier and M. Slatkin. Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Mol. Bio. Evolution*, 12:921–927, 1995.
- [9] 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.
- [10] M. Fullerton, A. Clark, K. Weiss, D. Nickerson, S. Taylor, J. Stengard, V. Salomaa, E. Var-
tinen, M. Perola, E. Boerwinkle, and C. Sing. Apolipoprotein E variation at the sequence
haplotype level: implications for the origin and maintenance of a major human polymorphism.
Am. J. of Human Genetics, 67(4):881–900, 2000.
- [11] D. Gusfield. Efficient algorithms for inferring evolutionary history. *Networks*, 21:19–28, 1991.
- [12] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational
Biology*. Cambridge University Press, 1997.
- [13] D. Gusfield. Inference of haplotypes from samples of diploid populations: complexity and
algorithms. *Journal of computational biology*, 8(3), 2001.
- [14] D. Gusfield. Haplotyping as Perfect Phylogeny: Conceptual Framework and Efficient Solu-
tions (Extended Abstract). In *Proceedings of RECOMB 2002: The Sixth Annual International
Conference on Computational Biology*, pages 166–175, 2002.

- [15] M. Hawley and K. Kidd. HAPLO: a program using the EM algorithm to estimate the frequencies of multi-site haplotypes. *J. Heredity*, 86:409–411, 1995.
- [16] L. Helmuth. Genome research: Map of the human genome 3.0. *Science*, 293(5530):583–585, 2001.
- [17] S. Lin, D. Cutler, M. Zwick, and A. Cahkravarti. Haplotype inference in random population samples. *Am. J. of Hum. Genet.*, 71:1129–1137, 2003.
- [18] T. Niu, Z. Qin, X. Xu, and J.S. Liu. Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *Am. J. Hum. Genet.*, 70:157–169, 2002.
- [19] S. Orzack, D. Gusfield, and V. Stanton. The absolute and relative accuracy of haplotype inferral methods and a consensus approach to haplotype inferral. Abstract Nr 115 in Am. Society of Human Genetics, Supplement 2001.
- [20] A. Piccolboni and D. Gusfield. Haplotyping by pure parsimony. Technical report, UC Davis, Department of Computer Science, 2003.
- [21] J. C. Stephens, J.A. Schneider, D.A. Tanguay, J. Choi, T. Acharya, S.E. Stanley, R. Jiang, C.J. Messer, A. Chew, J.H. Han, J. Duan, J.L. Carr, M.S. Lee, B. Koshy, A.M. Kumar, G. Zhang, W.R. Newell, A. Windemuth, C. Xu, T.S. Kalbfleisch, S.L. Shaner, K. Arnold, V. Schulz, C.M. Drysdale, K. Nandabalan, R.S. Judson, G. Ruano, and G.F. Vovis. Haplotype variation and linkage disequilibrium in 313 human genes. *Science*, 293:489–493, 2001.
- [22] 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.
- [23] S. Tavaré. 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, Washington DC, 1995.