PROC. 2ND RECOMB SWCM FOR SNPS AND HAPLOTYPES, IN LNBI, SPRINGER, 2004

On the Complexity of Haplotyping via Perfect Phylogeny

Jens Gramm, Till Nierhoff, Roded Sharan, and Till Tantau

International Computer Science Institute
1947 Center Street, Suite 600, Berkeley, CA 94704.
{gramm,nierhoff,roded,tantau}@icsi.berkeley.edu

Abstract. The problem of haplotyping via perfect phylogeny has received a lot of attention lately due to its applicability to real haplotyping problems and its theoretical elegance. However, two main research issues remained open: The complexity of haplotyping with missing data, and whether the problem is linear-time solvable. In this paper we settle the first question and make progress toward answering the second one. Specifically, we prove that Perfect Phylogeny Haplotyping with missing data is NP-complete even when the phylogeny is a path and only one allele of every polymorphic site is present in the population in its homozygous state. Our result implies the hardness of several variants of the missing data problem, including the general Perfect Phylogeny Haplotyping Problem with missing data, and Hypergraph Tree Realization with missing data. On the positive side, we give a linear-time algorithm for Perfect Phylogeny Haplotyping when the phylogeny is a path. This variant is important due to the abundance of yin-yang haplotypes in the human genome. Our algorithm relies on a reduction of the problem to that of deciding whether a partially ordered set has width 2.

1 Introduction

Single nucleotide polymorphisms (SNPs) are differences in a single base, across the population, within an otherwise conserved genomic sequence [10]. SNPs account for the majority of the variation between DNA sequences of different individuals. Especially when occurring in coding or otherwise functional regions, variations in the allelic content of SNPs are linked to medical conditions [18] or may affect drug response [15]. These examples underline the great clinical, scientific and commercial impact of efficient and accurate methods for SNP typing.

The sequence of alleles in contiguous SNP positions along a chromosomal region is called a *haplotype*. For diploid organisms, the *genotype* specifies for every SNP position the particular alleles which are present at this site in the two chromosomes. However, the genotype contains information only on the combination of alleles in a given site but not on the association of each allele with one of the two chromosomes. Current technology suitable for large-scale polymorphism screening obtains only the genotype information. The haplotypes for each chromosome can only be obtained at a considerably higher cost [14]. It is therefore desirable to develop efficient methods for inferring haplotypes from genotypes.

Known approaches for resolving haplotypes from genotype data include parsimony approaches [4], maximum likelihood methods [6], and statistical methods [16]. In this paper we address a perfect phylogeny-based technique for haplotype inference. Herein, the central idea is to resolve a given set of genotypes to haplotypes under the assumption that the haplotypes form a perfect phylogeny. This concept was introduced in a seminal paper by Gusfield [11]. The theoretical elegance of the perfect phylogeny approach to haplotyping as well as its efficiency and good performance in practice [3, 5] has spawned several studies of the problem and its variants [1, 5, 12]. In particular, quadratic-time algorithms have been given for the case of complete and error-free input data [1, 5]. Much of the current research around this problem focuses on two main questions: (1) 'Are there polynomial-time algorithms for variants of the problem that allow for missing data?'; and (2) 'Is the problem solvable in linear time?'.

In this paper we settle the first question and make progress toward answering the second one. Specifically, we prove that a restricted version of haplotyping via perfect phylogeny is NP-complete when missing data is allowed. In this newly introduced variant of the problem, only one allele of every genotyped SNP can be observed in its homozygous state; this is motivated by focusing on SNPs for which one of their alleles is lethal. Our result implies the NP-hardness of many variants of the missing data problem. Thus, we show the hardness of Perfect Phylogeny Haplotyping with missing data, both in the directed and in the undirected case, and even in the case that the phylogeny is a path. Further, we show, for the case of missing data, the hardness of Graph Realization and of a closely related problem called Perfect Phylogeny Xor Haplotyping. On the positive side, we give a linear-time algorithm for Perfect Phylogeny Haplotyping when the phylogeny is a path. This is the case in the presence of so-called *yin-yang haplotypes*, i.e., the particular haplotype pattern in which two haplotypes have different alleles at every SNP site. Recently, it has been discovered that yin-yang haplotypes span large parts of the human genome [19]. Our algorithm relies on a reduction of the problem to that of deciding whether a partially ordered set has width 2.

2 Preliminaries

A haplotype h is a binary string. A genotype g is a string over the alphabet $\{0, 1, 2\}$. For a string g let $g[i]$ denote the ith symbol of g. We say that a genotype $g \in \{0, 1, 2\}^m$ is *compatible* with the haplotypes $h^1, h^2 \in \{0, 1\}^m$ if for every i the following two conditions hold: (1) If $g[i] = 1$ or $g[i] = 0$ then $h^1[i] = h^2[i] = g[i]$; and (2) if $g[i] = 2$, then $h^1[i] \neq h^2[i]$. Let A be a $\{0, 1, 2\}$ genotype matrix of dimensions $n \times m$. A binary matrix B of dimensions $2n \times m$ is said to be *compatible* with A if for every i, row i of A is compatible with rows $2i - 1$ and $2i$ of B. For a matrix A let $A[i, j]$ denote the entry in row i and column j.

We say that A admits a *perfect phylogeny* if there exists a haplotype matrix B that is compatible with A and a rooted tree T_B such that:

1. Each column of B labels exactly one edge of T_B.
2. Every edge of T_B is labeled by at least one column of B.
3. Each row of B labels exactly one node of T_B.
4. For every row i of B the set of columns with value 1 in this row forms a path from T's root to the node labeled by i.

The basic problem that we study in this paper is the following:

Problem 2.1 (Perfect Phylogeny Haplotyping Problem, PPH).
Input: A genotype matrix A.
Question: Does A admit a perfect phylogeny?

The above problem is more precisely called the *directed* Perfect Phylogeny Haplotyping problem. In the directed case, the ancestral state of every SNP site is assumed to be 0 or, equivalently, the root corresponds to the all-0 haplotype. In the *undirected* case the ancestral state of every site can be arbitrary (0 or 1), and the columns on a path from the root to a node labeled by row i correspond to positions in which the value at row i differs from the corresponding ancestral state. We shall restrict attention to the directed case, but note that our hardness results apply also to the undirected case via a simple reduction that adds an all-0 row to the input matrix.

The definition of matrices A that admit a perfect phylogeny has the disadvantage that the properties of the tree are formulated in terms of the (generally unknown) matrix B, rather than in terms of A itself. A characterization that directly relates the tree to A can be derived from observations made by Gusfield [11]. This characterization is summarized in the following theorem.

Theorem 2.2. *A matrix A admits a perfect phylogeny iff there exists a rooted tree T_A such that:*

1. *Each column of A labels exactly one edge of T_A.*
2. *Every edge of T_A is labeled by at least one column of A.*
3. *For every row i of A:*
   (a) *The columns with value 1 in this row label a path from the root to some node u.*

(b) *The columns with value 2 in this row label a path that visits u and is contained in the subtree rooted at u.*

Proof. For the if-part, suppose that a tree T_A with the above properties exists. We construct a perfect phylogeny for A, consisting of a tree T_B and a haplotype matrix B. The topology of T_B is same as the topology of T_A. The edge labels are also the same. We assign node labels to T_B as follows: For each row i of A we place the labels $2i-1$ and $2i$ on two specific nodes v and v': These nodes are the end points of the path in T_B induced by the 2-entries in A in row i (possibly, these nodes coincide if the path is just the single node u). The haplotype matrix B can now easily be derived: For each row i in A we have two rows $2i-1$ and $2i$ in B. Each of these rows has a 1-entry exactly at those column positions that are on the path from the root to the nodes v or v', respectively.

For the only-if-part, suppose that a perfect phylogeny for A, consisting of a tree T_B and a haplotype matrix B, is given. We claim that the tree T_B, stripped of the node labels, is the desired tree T_A: Consider any row i of A and the two nodes v and v' to which the rows $2i-1$ and $2i$ of B are attached. The two paths p and p' leading from the root to v and v' are identical up to some node u. Then they split. Exactly in those columns corresponding to the edges on the path from the root to u, both row $2i-1$ and row $2i$ must have a 1-entry. On each column corresponding to an edge on the paths from u to v and from u to v', exactly one of the two rows must have the value 1. This shows that the columns in which A has a 1-entry in row i are the edges on the path from the root to u and that the columns in which A has a 2-entry in row i are the the edges on the path between v and v'. This path contains u. □

Given a genotype matrix A, we refer to the tree T_A with the labeling as described in Theorem 2.2 as *perfect phylogeny tree* for A. The following lemma is a useful observation that follows easily from the properties of T_A.

Lemma 2.3. *Let A be a genotype matrix that admits a perfect phylogeny tree T_A. Consider a path starting at the root of T_A and let A' be the matrix that consists of the columns that label this path, in the order in which they appear on the path. Then each row of A' is of the form* $(1,\dots,1,2,\dots,2,0,\dots,0)$. □

We now define several variants of PPH, which we study in the sequel. In *Perfect Phylogeny Path Haplotyping (PPPH)* one has to determine if the input genotype matrix admits a perfect phylogeny that is a path. Such a problem arises for instances that contain an all-2 row, corresponding to yin-yang haplotypes which were shown to be common in human populations [19]. For convenience, we use an equivalent formulation of PPPH:

Problem 2.4 (Perfect Phylogeny Path Haplotyping, PPPH).
Input: A genotype matrix A containing an all-2 row.
Question: Does A admit a perfect phylogeny?

The input matrix may contain missing entries, manifested as question mark entries. A matrix with missing entries is called *incomplete*. This leads to the following problem:

Problem 2.5 (Perfect Phylogeny Haplotyping with missing entries).
Input: An incomplete genotype matrix A.
Question: Is there a completion A' of the missing entries in A such that the resulting matrix admits a perfect phylogeny?

Analogously, we can define Perfect Phylogeny Path Haplotyping with missing entries. In this work, we also consider the following variants of Perfect Phylogeny (Path) Haplotyping (with missing entries): In one variant, we consider only SNPs for which one of their alleles (denoted here by 1) is lethal and, therefore, not observed in its homozygous state in the population. Therefore, the input matrix (and possible completions thereof) contain only entries 0 and 2. A second variant is motivated by data in which we only have information on whether a SNP state is homozygous or heterozygous, i.e., for a homozygous state we do not know which of the two alleles is present. For complete data, this problem is called *Xor Perfect Phylogeny Haplotyping (XPPH)*, and was shown to be equivalent to Hypergraph Tree Realization [2].

Notation. In the following we use a uniform notation for all problem variants that we consider, indicating in each case the allowed values in the input matrix: Thus, we use $\{0, 1, 2\}$-PPH ($\{0, 1, 2\}$-PPPH) to denote the Perfect Phylogeny (Path) Haplotyping Problem and $\{0, 1, 2, ?\}$-PPH ($\{0, 1, 2, ?\}$-PPPH) to denote the corresponding version with missing data.

3 Hardness Results

3.1 Perfect Phylogeny Path Haplotyping with Missing Data

In this section we study the complexity of $\{0, 2, ?\}$-PPPH which is the version of Perfect Phylogeny Path Haplotyping with missing data for which, in the input matrix and possible completions thereof, only entries 0 and 2 are allowed. We show NP-hardness of $\{0, 2, ?\}$-PPPH by giving a reduction from the NP-complete Not-All-Equal 3SAT (NAE3SAT) [8]. Given a Boolean formula in conjunctive normal form with three literals per clause, for NAE3SAT we must decide whether there is an assignment to the variables such that in every clause at least one and at most two literals are satisfied.

Theorem 3.1. *$\{0, 2, ?\}$-PPPH is NP-complete.*

Proof. We first outline the reduction and then show its correctness.

Reduction. Let F be a 3-CNF formula over variables $v_1, v_2, \ldots, v_n$ and clauses $C_1, C_2, \ldots, C_m$. Each clause C_j is given as a set of three literals $\{l_{j,1}, l_{j,2}, l_{j,3}\}$, where each literal $l_{j,r}$ is either a variable v_i or a negated variable $\bar{v}_i$.

We map F to a matrix A with entries from $\{0, 2, ?\}$ with $2n + 3m + 2$ rows and $2n + 3m$ columns. The construction proceeds columnwise. For $x \in \{0, 2, ?\}$ and a positive integer i let x^i denote a length-i column vector containing only entries x.

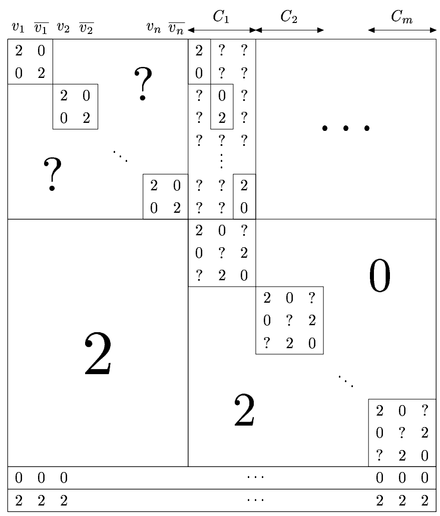

Fig. 1. Illustration of the matrix A from the proof of Theorem 3.1. It is constructed from a 3-CNF formula with variables $v_1, v_2, \ldots, v_n$ and clauses $C_1, C_2, \ldots, C_m$. In the above example $C_1 = \{v_1, \bar{v}_2, v_n\}$.

Encoding literals. For each variable v_i we define two column vectors $\langle v_i \rangle$ and $\langle \bar{v}_i \rangle$:

$$\langle v_i \rangle := \begin{pmatrix} ?^{2(i-1)} \\ 2 \\ 0 \\ ?^{2(n-i)} \end{pmatrix} \text{ and } \langle \bar{v}_i \rangle := \begin{pmatrix} ?^{2(i-1)} \\ 0 \\ 2 \\ ?^{2(n-i)} \end{pmatrix}$$

Encoding clauses. For each clause $C_j = \{l_{j,1}, l_{j,2}, l_{j,3}\}$ with $j \in \{1, \ldots, m\}$ we define the following three column vectors:

$$\langle C_{j,1} \rangle := \begin{pmatrix} \langle l_{j,1} \rangle \\ 0^{3(j-1)} \\ 2 \\ 0 \\ ? \\ 2^{3(m-j)} \end{pmatrix}, \langle C_{j,2} \rangle := \begin{pmatrix} \langle l_{j,2} \rangle \\ 0^{3(j-1)} \\ 0 \\ ? \\ 2 \\ 2^{3(m-j)} \end{pmatrix}, \langle C_{j,3} \rangle := \begin{pmatrix} \langle l_{j,3} \rangle \\ 0^{3(j-1)} \\ ? \\ 2 \\ 0 \\ 2^{3(m-j)} \end{pmatrix}.$$

Resulting matrix. The matrix A is composed of the following columns: For every variable v_i it contains the two *variable columns* $\binom{\langle v_i \rangle}{2^{3m}}$ and $\binom{\langle \bar{v}_i \rangle}{2^{3m}}$. For each clause C_j it contains the three *clause columns* $\langle C_{j,1} \rangle$, $\langle C_{j,2} \rangle$, and $\langle C_{j,3} \rangle$. Below these columns, we add an all-0 *row* and an all-2 *row* to the matrix. The all-0 row is added to enforce a directed perfect phylogeny, the all-2 row is added to enforce that the resulting perfect phylogeny takes the form of a path. The resulting matrix is illustrated in Figure 1.

Correctness. For the correctness of the reduction, we show that a 3-CNF formula F has a satisfying assignment such that in each clause exactly one or exactly two literals are satisfied iff the matrix A has a completion A' that admits a perfect phylogeny.

Only-if-part. Let a satisfying assignment $\tau\colon \{v_1, \ldots, v_n\} \to \{0,1\}$ be given. We extend it to an assignment that assigns values also to the negated literals. We show how to complete the matrix A to a matrix A' that admits a perfect phylogeny.

First, consider a variable column c corresponding to a literal l with $l = v_i$ or $l = \bar{v}_i$. For every variable v_h with $h \neq i$ we replace the question marks in the positions $A'[2h-2, c]$ and $A'[2h-1, c]$ by $\binom{2}{0}$ if $\tau(l) = \tau(v_h)$, and by $\binom{0}{2}$ otherwise.

Second, consider a clause column c corresponding to a literal $l_{j,r}$ in a clause C_j. Let $l_{j,r}$ equal v_i or $\bar{v}_i$. Again, for every variable v_h with $h \neq i$ we replace the question marks in the positions $A'[2h-2, c]$ and $A'[2h-1, c]$ by $\binom{2}{0}$ if $\tau(l_{j,r}) = \tau(v_h)$, and by $\binom{0}{2}$ otherwise.

For the completion of the lower part of the clause columns, consider a clause C. It contains three literals l_1, l_2, and l_3. Exactly two of these must be false under the assignment τ or exactly two of them must be true. If l_1 and l_2 are these two literals, we set the question mark in l_1's column to a 2. If they are l_2 and l_3, we set the question mark in l_2's column to a 2. Finally, for l_3 and l_1 we set the question mark in l_3's column to a 2. All remaining question marks are set to 0.

It remains to show that the resulting matrix A' has a perfect phylogeny. For this, we construct a perfect phylogeny tree $T_{A'}$ for A'. Since A' contains a row of 2-entries, this tree must, due to Theorem 2.2, actually take the form of a (rooted) path. We call the subpath of $T_{A'}$ to one side of the root the *true side* of $T_{A'}$ and call the subpath to the other side of $T_{A'}$ the *false side*. Starting from the root node, the path $T_{A'}$ is constructed edge by edge as follows: Firstly, in order of increasing index i, we consider literals v_i and $\bar{v}_i$. If v_i evaluates to true w.r.t. τ, then we add an v_i-edge on the true side of $T_{A'}$, i.e., we add an edge on the true side and label this edge with the variable column corresponding to v_i; in the same way, we add a $\bar{v}_i$-edge on the false side of $T_{A'}$. If v_i evaluates to false w.r.t. τ, we add an v_i-edge on the false side and an $\bar{v}_i$-edge on the true side. Secondly, again in order of increasing index i, we consider each clause C_i. Let it contain the three literals l_1, l_2, and l_3. For those literals of l_1, l_2, and l_3 that evaluate to true w.r.t. τ, we add an edge on the true side of $T_{A'}$ and for those literals that evaluate to false w.r.t. τ, we add an edge on the false side of $T_{A'}$ (in each case labeled by the corresponding clause column). Exactly two of these three clause columns will be placed on the same side. These two columns are internally ordered as follows: we maintain the order of l_1, l_2, and l_3, except in the situation when the l_1-edge and the l_3-edge are added on the same side of $T_{A'}$; then, we place the l_3-edge closer to the root of $T_{A'}$ than the l_1-edge.

We claim that the resulting path $T_{A'}$ is a perfect phylogeny tree for A'. We can show this by testing that $T_{A'}$ with its indicated edge labeling satisfies the conditions of Theorem 2.2. Since A' does not contain 1-entries, it remains to test condition 3(b). We only sketch here how this test is done: First, consider a row of A' in the upper part. Then the columns in which this row has value 2 will form a complete path from the root to one of the ends of T_A. Second, consider any three rows of A' in the lower part corresponding to a clause C. In all three rows the set of columns in which this row has value 2 forms a path inside T_A; this path extends between two clause columns which correspond to two literals of C and which are placed on different sides of T_A (including these columns or stopping just before them).

If-part. Let A' be a completion of A such that A' admits a perfect phylogeny. Let $T_{A'}$ be the perfect phylogeny tree for A'. Since A contains an all-2 column, $T_{A'}$ has to take the form of a path. We may assume that each edge is labeled with exactly one column of A'. Let us say that the path has two *sides*, namely the edges leading from the root to one end of the path and the edges leading from the root to the other end. We extract an assignment $\tau\colon \{v_1, \dots, v_n\} \to \{0, 1\}$ from $T_{A'}$ as follows: Choose any side of $T_{A'}$ and let E be the set of edges on this side. Let $\tau(v) = 1$ iff v's column labels an edge in E. We claim that the resulting assignment satisfies the formula F which follows from these observations:

1. The variable columns of a literal v and the literal $\bar{v}$ must be on different sides of $T_{A'}$. This follows with Lemma 2.3 from the $\left(\begin{smallmatrix} 2 & 0 \\ 0 & 2 \end{smallmatrix}\right)$ block in these columns.
2. A clause column of a literal l in any clause must be on the same side of $T_{A'}$ as the variable column of l. This follows with Lemma 2.3 from the $\left(\begin{smallmatrix} 2 & 0 \\ 0 & 2 \end{smallmatrix}\right)$ block formed by the upper part of l's clause column and the variable column corresponding to $\bar{l}$. Thus the clause column for l and the variable column for $\bar{l}$ must be on different sides and hence, with (1), the clause column for l and the variable column for l must be on the same side.
3. For any clause $C = \{l_1, l_2, l_3\}$ the clause columns corresponding to these literals cannot all be on the same side of $T_{A'}$. This is due to the $\left(\begin{smallmatrix} 2 & 0 & ? \\ 0 & ? & 2 \\ ? & 2 & 0 \end{smallmatrix}\right)$ block, of A which cannot be completed in such a way that all three columns are on the same side of $T_{A'}$; this can be shown by trying all possible completions and using Lemma 2.3. □

3.2 Implications for Perfect Phylogeny Haplotyping Problems

In the previous section we have shown that $\{0, 2, ?\}$-PPPH, a restricted case of Perfect Phylogeny Haplotyping with missing data, is NP-hard. In the following we use this result to show the hardness of several related variants of Perfect Phylogeny Haplotyping with missing data:

Theorem 3.2. *$\{0, 2, ?\}$-PPH, $\{0, 1, 2, ?\}$-PPPH, and $\{0, 1, 2, ?\}$-PPH are NP-complete.*

Proof. The Perfect Phylogeny *Path* Haplotyping problem is a subproblem of Perfect Phylogeny Haplotyping. Therefore the hardness of $\{0, 2, ?\}$-PPPH implies the hardness of $\{0, 2, ?\}$-PPH, and the hardness of $\{0, 1, 2, ?\}$-PPPH implies the hardness of $\{0, 1, 2, ?\}$-PPH. Therefore it remains to show that $\{0, 1, 2, ?\}$-PPPH is NP-hard. To this end, we use the following reduction from $\{0, 2, ?\}$-PPPH to $\{0, 1, 2, ?\}$-PPPH: Let A be an input instance for $\{0, 2, ?\}$-PPPH. We map it to an instance $\hat{A} = A$ for $\{0, 1, 2, ?\}$-PPPH. Clearly a solution to A is also a solution to the $\{0, 1, 2, ?\}$-PPPH instance. Conversely, let $\hat{A}'$ be a completion of

$\hat{A}$ that admits a perfect phylogeny. Let $T_{\hat{A}}$ be the perfect phylogeny tree for $\hat{A}$ which takes the form of a path due to Theorem 2.2. Consider the matrix C that is obtained from $\hat{A}'$ by replacing every 1-entry with a 2-entry and let T_C be the tree obtained from $T_{\hat{A}}$ by replacing the column labels from $\hat{A}$ with the corresponding columns in C. Obviously, C is also a completion for A with only 0-entries and 2-entries, T_C satisfies the conditions of Theorem 2.2, and, thus, C admits a perfect phylogeny. □

We note that the NP-completeness of $\{0,1,2,?\}$-PPH was independently shown by Kimmel [13].

3.3 Implications for Hypergraph Realization Problems

Gusfield has shown that Perfect Phylogeny Haplotyping reduces to the classical Hypergraph Tree Realization Problem [11]. This problem was later shown to be equivalent to XPPH [2]. In this section we show that Hypergraph Tree Realization and restricted variants thereof are NP-hard in the presence of missing data. In particular, we strengthen a result of Golumbic and Wassermann [9] for hypergraphs. In this context, the 'missing data scenario' is formulated as a 'sandwich' problem of two hypergraphs as follows.

A *hypergraph* is a pair $H = (V, E)$ consisting of a vertex set V and a set E of subsets of V. The elements of E are called *hyperedges*. For a *Hypergraph Sandwich Problem* we are given a pair $H^1 = (V, E^1)$ and $H^2 = (V, E^2)$ of hypergraphs as input such that $E^1 = \{e^1_1, \dots, e^1_m\}$ and $E^2 = \{e^2_1, \dots, e^2_m\}$ with $e^1_i \subseteq e^2_i$ for all $i \in \{1, \dots, m\}$. The goal is to find a hypergraph $H = (V, E)$ that is 'sandwiched' between H^1 and H^2, that is, $E = \{e_1, \dots, e_m\}$ with $e^1_i \subseteq e_i \subseteq e^2_i$ for all $i \in \{1, \dots, m\}$.

Problem 3.3 (Interval Hypergraph Sandwich Problem).
Input: Two hypergraphs $H_1 = (V, E_1)$, $H_2 = (V, E_2)$.
Question: Is there a hypergraph H sandwiched between H^1 and H^2 and a linear ordering of V such that each hyperedge $e \in E$ is an interval?

Golumbic and Wassermann have shown that this problem is NP-complete [9]. We prove the following, stronger theorem.

Theorem 3.4. *The Interval Hypergraph Sandwich Problem is* NP*-complete, even if we require that all hyperedges in E_1 share a common vertex.*

Proof. (Sketch) We describe a reduction from $\{0,2,?\}$-PPPH to this problem. Its correctness is omitted due to lack of space. Let A be an input matrix to $\{0,2,?\}$-PPPH. For each row i of A let $e^1_i = \{c \mid A[i,c] = 2\}$ and $e^2_i = e^1_i \cup \{c \mid A[i,c] = ?\}$ The reduction maps A to $H^1 = (V, E^1)$ and $H^2 = (V, E^2)$, where V is the set of columns of A, $E^1 = \{e^1_1, e^1_2, \dots\}$, and $E^2 = \{e^2_1, e^2_2, \dots\}$. □

Our hardness result also implies the hardness of the general Hypergraph Tree Realization Problem with missing data:

Problem 3.5 (Hypergraph Tree Realization Sandwich Problem).
Input: Two hypergraphs $H^1 = (V, E^1)$, $H^2 = (V, E^2)$.
Question: Are there a hypergraph H sandwiched between H^1 and H^2 and a tree T whose set of edges is V such that every hyperedge $e \in E$ is a path in T?

Using the same reduction as in the proof of Theorem 3.4, we conclude:

Theorem 3.6. *The Hypergraph Tree Realization Sandwich Problem is* NP*-complete, even if we require that all hyperedges in E_1 share a common vertex.* □

4 A Linear-Time Algorithm for PPPH

We complement the hardness results given in the previous sections with an algorithmic result. Several polynomial-time algorithms exist for $\{0,1,2\}$-PPH, but none is linear [1, 5, 11]. The running time of the algorithms is still superlinear if they are applied 'as is' to $\{0,1,2\}$-PPPH. In the following, we describe an algorithm that solves $\{0,1,2\}$-PPPH in linear time.

Theorem 4.1. *There exists a linear-time algorithm for solving* $\{0,1,2\}$-PPPH.

Proof. We are given an $n \times m$ genotype matrix A as an input for $\{0,1,2\}$-PPPH. We describe an algorithm that computes a perfect phylogeny tree T_A for A, if it exists, and, otherwise, reports that A does not admit a perfect phylogeny. *Preliminaries.* Let C be the set of columns of A. We define a partial order $\succeq$ on C as follows: *c dominates c'* ($c \succeq c'$) iff, for every i, $c(i) \succeq c'(i)$, where $1 \succ 2 \succ 0$. If A admits a perfect phylogeny tree T_A, then each path starting at the root is, by Lemma 2.3, a chain in $(C, \succeq)$. A perfect phylogeny tree for A has, due to the all-2 row in A and Theorem 2.2, to take the form of a (rooted) path. Therefore, $(C, \succeq)$ has a cover by two chains, i.e. has width at most two. Initially, the algorithm computes for every column of A its *leaf count*, which is twice the number of 1-entries plus the number of 2-entries in this column. The following two phases of the algorithm rely on these leaf counts: In its first phase, the algorithm computes those edges in T_A that are labeled by a column containing a 1-entry. These edges form a path that is called the *initial* path of the matrix A and it was already shown by Gusfield [11] that this initial path can be computed in linear time. In its second phase, the algorithm expands the initial path to T_A by processing those columns in A that do not contain a 1-entry. Phase 2 is based on an algorithm by Felsner, Raghavan, and Spinrad [7] for checking in linear time whether a partially ordered set (poset) has width 2. In the following, we describe these two phases in more detail.
Phase 1. We consider the columns of A containing a 1-entry. Let $G = (C, E)$ be the graph, where $\{c, c'\} \in E$ iff there is a row in A which has a 1-entry both in c and c'. Assume that A admits a perfect phylogeny and let $c, c' \in C$. If $\{c, c'\} \in E$, then Theorem 2.2, condition 3(a), implies that c and c' must be on the same path starting at the root. Otherwise, Lemma 2.3 implies that they are on different paths. It follows that G consists of one or two cliques. As

observed by Gusfield [11], in any perfect phylogeny tree for A the leaf counts are decreasing on any path down from the root. The algorithm chooses an arbitrary column c with a 1-entry. Then it computes the set C_1 of c and all its neighbors in G. We obtain two sets: C_1 and the set C_2 containing the remaining columns with a 1-entry. The algorithm tests whether each set, ordered by the leaf count, is a chain with respect to $\succeq$. If not, it is safe to stop and output that A does not admit a perfect phylogeny. Otherwise, the initial path is given by the two chains.

Phase 2. Ordering the columns by their leaf count gives a linear extension (for terminology related to posets see, e.g., the book by Trotter [17]) of the poset $(C, \succeq)$. Given that linear extension, an algorithm by Felsner, Raghavan, and Spinrad [7, Thm 3] decides in time $O(mn)$ whether the width of $(C, \succeq)$ is at most two. We use a straightforward modification of their algorithm to produce (in the same time bound) a Hasse diagram of $(C, \succeq)$ of width two, if existent; otherwise, we terminate with the output that A does not admit a perfect phylogeny. For a width-2 Hasse diagram, let c and c' be the ends of the initial path. The columns without 1-entries are entirely below c and c' in the Hasse diagram. We extend the initial path as follows: We append to c a maximal chain below c (by appending a path containing one edge for every column in the chain, in the same order as they appear in the chain). In the same way, we append all remaining columns (another chain, not necessarily maximal) to c'. The resulting rooted path is the output of the algorithm.

Correctness. Whenever the algorithm reports that A does not admit a perfect phylogeny, then this is correct, as argued in the description. Otherwise, the (rooted) path returned by the algorithm satisfies the properties stated in Theorem 2.2 and is thus in fact a perfect phylogeny for A.

Running time. Computing the leaf counts can be done in linear time. Likewise can the column sets C_1 and C_2 be determined in linear time. Since the leaf counts range only between 0 and $2n$, the set C can be ordered according to the leaf count in linear time. Using that order, the initial path can be built out of C_1 and C_2 in linear time. The Hasse diagram has at most $2m$ edges. Therefore, the traversals in Phase 2 can also be done in linear time. □

Acknowledgments. We would like to thank Eran Halperin for insightful discussions on haplotyping with missing data. J.G., T.N. and T.T. were supported through a postdoc fellowship by the DAAD (German Academic Exchange Service). This research was supported in part by NSF ITR Grant CCR-0121555.

References

1.
2. T. Barzuza, J. S. Beckmann, R. Shamir, and I. Pe'er. Xor haplotyping: Resolution of perfect phylogeny haplotypes from heterozygote/homozygote calls. Manuscript in preparation, December 2003.

3. R. H. Chung and D. Gusfield. Empirical exploration of perfect phylogeny haplotyping and haplotypers. In *Proc. of the 9th COCOON*, number 2697 in LNCS, pages 5–19. Springer, 2003.
4. A.G. Clark. Inference of haplotypes from PCR-amplified samples of diploid populations. *Journal of Molecular Biology and Evolution*, 7(2):111–22, 1990.
5. E. Eskin, E. Halperin, and R. M. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. *Journal of Bioinformatics and Computational Biology*, 1(1):1–20, 2003.
6. L. Excoffier and M. Slatkin. Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Molecular Biology and Evolution*, 12(5):921–7, 1995.
7. S. Felsner, V. Raghavan, and J. Spinrad. Recognition algorithms for orders of small width and graphs of small Dilworth number. Technical Report TR-B-99-05, FU Berlin, 1999. Available from http://www.math.tu-berlin.de/~felsner/Paper/width.ps.gz. To appear in *Order*.
8. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman, San Francisco, 1979.
9. M. C. Golumbic and A. Wassermann. Complexity and algorithms for graph and hypergraph sandwich problems. *Graphs and Combinatorics*, 14:223–9, 1998.
10. The International SNP Map Working Group. A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. *Nature*, 409:928–33, 2001.
11. D. Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. In *Proc. of 6th RECOMB*, pages 166–75. ACM Press, 2002.
12. E. Halperin and R. M. Karp. Perfect phylogeny and haplotype assignment. Technical Report TR-678-03, Princeton University, October 2003.
13. G. Kimmel. The incomplete perfect phylogeny haplotype problem. Technical report, School of Computer Science, Tel-Aviv University, 2003.
14. N. Patil, A.J. Berno, D.A. Hinds, et al. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, 294(5547):1719–23, 2001.
15. A. D. Roses. Pharmacogenetics and the practice of medicine. *Nature*, 405:857–65, 2000.
16. M. Stephens, N. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction from population data. *American Journal of Human Genetics*, 68(4):978–89, 2001.
17. W. T. Trotter. *Combinatorics and Partially Ordered Sets: Dimension Theory.* The Johns Hopkins University Press, Baltimore, 1992.
18. Y. Watanabe, A. Fujiyama, Y. Ichiba, M. Hattori, et al. Chromosome-wide assessment of replication timing for human chromosomes 11q and 21q: disease-related genes in timing-switch regions. *Human Molecular Genetics*, 11(1):13–21, 2002.
19. J. Zhang, W. L. Rowe, A. G. Clark, and K. H. Buetow. Genomewide distribution of high-frequency, completely mismatching SNP haplotype pairs observed to be common across human populations. *American Journal of Human Genetics*, 73(5):1073–81, 2003.