Available online at www.sciencedirect.com
ScienceDirect
Discrete Applied Mathematics 155 (2007) 788–805
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Haplotyping with missing data via perfect path phylogenies☆

Jens Gramm[a,1,2,3], Till Nierhoff[b,2], Roded Sharan[c,1,4], Till Tantau[d,*,1,2]

[a] *Wilhelm-Schickard Institut für Informatik, Universität Tübingen, Germany*
[b] *International Computer Science Institute, Berkeley, USA*
[c] *School of Computer Science, Tel-Aviv University, Israel*
[d] *Institut für Theoretische Informatik, Univeristät zu Lübeck, Germany*

Received 16 July 2004; accepted 29 September 2005
Available online 13 October 2006

Abstract

Computational methods for inferring haplotype information from genotype data are used in studying the association between genomic variation and medical condition. Recently, Gusfield proposed a haplotype inference method that is based on perfect phylogeny principles. A fundamental problem arises when one tries to apply this approach in the presence of missing genotype data, which is common in practice. We show that the resulting theoretical problem is NP-hard even in very restricted cases. To cope with missing data, we introduce a variant of haplotyping via perfect phylogeny in which a *path* phylogeny is sought. Searching for perfect path phylogenies is strongly motivated by the characteristics of human genotype data: 70% of real instances that admit a perfect phylogeny also admit a perfect path phylogeny. Our main result is a fixed-parameter algorithm for haplotyping with missing data via perfect path phylogenies. We also present a simple linear-time algorithm for the problem on complete data.

MSC: 68W05; 92-04; 92-08; 92D10; 92D15; 92D20

Keywords: Haplotyping; Haplotypes; Genotypes; Missing data; Incomplete data; Phylogenetics; Perfect phylogenies; Path phylogenies; Fixed-parameter algorithms

1. Introduction

Single nucleotide polymorphisms (SNPs) are differences in a single base, across the population, within an otherwise conserved genomic sequence [21]. SNPs account for the majority of the variation between DNA sequences of different

☆ Portions of this paper appeared in [12,13].

* Corresponding author.
E-mail addresses: gramm@informatik.uni-tuebingen.de (J. Gramm), nierhoff@icsi.berkeley.edu (T. Nierhoff), roded@tau.ac.il (R. Sharan), tantau@tcs.uni-luebeck.de (T. Tantau).

[1] Work done in part at the International Computer Science Institute, Berkeley.

[2] Supported by DAAD (German academic exchange service) postdoc research fellowship grants.
[3] Supported by DFG (German research association) Grant NI 369/2.
[4] Supported by an Alon fellowship.

0166-218X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2005.09.020

individuals [19]. Especially when occurring in coding or otherwise functional regions, variations in the allelic content of SNPs are linked to medical condition or may affect drug response.

The sequence of alleles in contiguous SNP positions along a chromosomal region is called a *haplotype*. A SNP commonly has two variants, or *alleles*, in the population, corresponding to two of the four genomic letters A, C, G, and T. For diploid organisms, the *genotype* specifies for every SNP position the particular alleles that are present at this site in the two chromosomes. Genotype data contains information only on the combination of alleles at a given site, it does not reveal the association of each allele with one of the two chromosomes. Current technology, suitable for large-scale polymorphism screening, obtains only the genotype information at each SNP site. The actual haplotypes in the typed region can be obtained at a considerably higher cost [19]. Due to the importance of haplotype information in association studies, it is desirable to develop efficient methods for inferring haplotypes from genotype information.

Extant approaches for resolving haplotypes from genotype data include parsimony approaches [5,14], maximum likelihood methods [8], and statistical methods [18,20]. In this paper we study a perfect-phylogeny-based technique for haplotype inference, first introduced in a seminal paper by Gusfield [15]. This approach assumes that the underlying haplotypes can be arranged in a phylogenetic tree, so that for each SNP site the set of haplotypes with the same state at this site forms a connected subtree. Such an assumption is particularly appropriate for short genomic regions that have not undergone recombination events. For longer regions, it is common practice to sidestep the recombination problem by inferring haplotypes only for small blocks of data and then assembling these blocks to obtain the complete haplotypes [7].

The theoretical elegance of the perfect phylogeny approach to haplotyping as well as its efficiency and good performance in practice [3,6] have spawned several studies of the problem and its variants [1,6,16]. In particular, quadratic-time algorithms have been devised for the case of complete input data [1,6].

A fundamental problem in applying this haplotyping approach in practice is the need to deal with missing data. Real genotype data usually contain a small fraction of missing entries caused by technical problems in the process of genotype detection. The perfect phylogeny haplotyping problem with missing data calls for finding a completion of the input genotypes that admits a perfect phylogeny. Previous work on this problem by Halperin and Karp [16] gives a polynomial algorithm for the case that the input genotypes satisfy the "rich-data hypothesis" (for any two SNPs one observes exactly three out of four possible state combinations). The complexity of the problem for general inputs remained open.

In this paper, we prove that the problem of perfect phylogeny haplotyping with missing data is NP-hard even under very restrictive assumptions. Part of this result was independently obtained by Kimmel and Shamir [17]. On the positive side, we introduce a novel approach to solve the perfect phylogeny haplotyping problem in the case of missing data. Our approach is motivated by the observation that more than two thirds of the human genome can be covered by *yin-yang haplotypes*, which are haplotype pairs that are heterozygous at every SNP site [16]. In the perfect phylogeny model for haplotyping, the presence of yin-yang haplotypes implies that any phylogeny has to take the form of a path. Therefore, we consider a variant of perfect phylogeny haplotyping in which one searches for underlying haplotypes that form a perfect *path* phylogeny (a phylogeny with at most two leaves). For the case of complete input data, we give an algorithm that solves the problem in linear time. For the case of incomplete data, this version of the problem is still NP-hard, but we show that there exists a fixed-parameter algorithm for it with respect to the maximum number of missing entries per SNP site. Our methods rely on a connection between the existence of a perfect path phylogeny and the width of a certain partial order on the columns of the genotype matrix.

To determine the abundance of path phylogenies in genotype data we examined an extensive collection of real genotype data sets. Our results show that in 70% of all cases in which the input matrix admits a perfect phylogeny, it also admits a perfect path phylogeny. In contrast, the rich-data hypothesis applies only to 23% of these cases. In particular, a large fraction of the data allows perfect path phylogenies but fails to satisfy the rich-data hypothesis. These results demonstrate that our methods can cope with the majority of available genotype data.

The paper is organized as follows: In Section 2, we introduce the perfect phylogeny haplotyping problem and the variants that are studied in this paper. In Section 3, we show that perfect phylogeny haplotyping with missing data and restrictive variants of it are NP-hard. In Section 4, we outline the connections between perfect phylogeny haplotyping and poset theory. Building on this theoretical framework, in Section 5 we present a linear-time algorithm for the problem when the target phylogeny is a path. Finally, in Section 6 we show that in the presence of missing data, perfect path phylogeny haplotyping is fixed-parameter tractable with respect to the maximum number of missing entries per SNP site.

2. Problem statement

When stripped of the biological context, the haplotype inference problem is a purely combinatorial problem, which we describe in this section. We concentrate on bi-allelic SNPs, as sites with more alleles are rare. In the combinatorial setting, a *haplotype* is a row vector with binary entries. Each position of the vector corresponds to a SNP site. When we observe a certain base at the SNP site, the vector contains a 0-entry at the corresponding position; if we observe a certain other base, the vector contains a 1-entry. For a haplotype h, let $h[i]$ denote the ith position of h. A *haplotype matrix* is a binary matrix whose rows are haplotypes.

A *genotype* is a row vector with entries in $\{0, 1, 2\}$, each corresponding to a SNP site. A 0- or 1-entry in a genotype implies that the two underlying haplotypes have the same entry in this position. A 2-entry in a genotype implies that the two underlying haplotypes differ at that position. A *genotype matrix* is a matrix whose rows are genotypes. Two haplotypes h_1 and h_2 *explain* a genotype g if for each position i the following holds: $g[i] \in \{0, 1\}$ implies $h_1[i] = h_2[i] = g[i]$; and $g[i] = 2$ implies $h_1[i] \neq h_2[i]$. Given an $n \times m$ genotype matrix A and a $2n \times m$ haplotype matrix B, we say that B *explains* A iff for every $i \in \{1, \ldots, n\}$ the haplotypes in rows $2i - 1$ and $2i$ of B explain the genotype in row i of A. For a genotype g and a value $v \in \{0, 1, 2\}$, the set of columns with value v in g is called the v-*set* of g.

2.1. Perfect phylogeny haplotyping

The following definition of a haplotype matrix that admits a perfect phylogeny is adapted from [1].

Definition 1. We say that *a haplotype matrix B admits a perfect phylogeny* if there exists a rooted tree T_B such that:

(1) Every row of B labels exactly one leaf of T_B, and each leaf is labeled by one row.
(2) Each column of B labels exactly one edge of T_B.
(3) Every interior edge of T_B is labeled by at least one column of B.
(4) For every two rows h_1 and h_2 of B and every column i, we have $h_1[i] \neq h_2[i]$ iff i lies on the path from h_1 to h_2 in T_B.

The tree T_B can be made more compact by placing the haplotypes also in interior nodes, rather than only at the leaves. In this case, condition (1) in the definition above is replaced with the requirement that every row of B labels exactly one node of T_B; and condition (3) is extended to all edges of T_B. We use this compact representation in the sequel.

Given an $n \times m$ genotype matrix A, we say that it *admits a perfect phylogeny* if there is a $2n \times m$ haplotype matrix B that explains A and admits a perfect phylogeny. The basic problem that we study in this paper is the following:

Problem 2 (*Perfect Phylogeny Haplotyping,* $\{0, 1, 2\}$-PPH).
Input: *A genotype* $\{0, 1, 2\}$-*matrix* A.
Question: *Does* A *admit a perfect phylogeny*?

In general, the haplotype labeling the root of a perfect phylogeny tree can have arbitrary ancestral states (0 or 1) at each site. In the *directed* version of perfect phylogeny haplotyping the ancestral state of every SNP site is assumed to be 0 or, equivalently, the root of the phylogenetic tree corresponds to the all-0 haplotype. As shown by Eskin et al. [6], one can reduce $\{0, 1, 2\}$-PPH to the directed case using a simple transformation of the input matrix: In each column of the genotype matrix search for the first non-2-entry from above; and if this entry is a 1-entry, exchange the roles of 0-entries and 1-entries in this column.

We now state an important characterization of directed perfect phylogenies, which is implicit in [15]:

Theorem 3. *A genotype matrix A admits a directed 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.*

A's columns	1	2	3	4	5	6	7
row *a*	**0**	**1**	**2**	**2**	**0**	**0**	**2**
row b	0	1	1	0	2	2	0
row c	0	1	0	2	0	0	2
row d	2	2	0	0	0	0	0

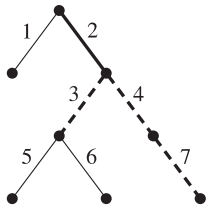

Fig. 1. An example of a genotype matrix A and a directed perfect phylogeny T_A for A. The edges of T_A are labeled by the columns of A. The paths induced by the first row a of A are shown in bold. The path corresponding to the 1-entries in row a leads from the root to an inner vertex of the tree (solid); the path corresponding to the 2-entries is rooted at this inner vertex (dashed). The end points of the dashed path are the nodes to which the two haplotypes 0110000 and 0101001 of row a are assigned in the tree T_B.

(3) *For every row r of A*:
  (a) *The columns in the* 1*-set of row r label a path from the root to some node u*.
  (b) *The columns in the* 2*-set of row r label a path that visits u and is contained in the subtree rooted at u*.

Proof. *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 and its edge labels are the same as those of T_A. 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 of A in row i (possibly, these nodes coincide if the path is just a single node). The haplotype matrix B can now be derived as follows: 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 w, respectively.

Only-If-Part: Suppose we are given a directed perfect phylogeny for A, consisting of a tree T_B and a haplotype matrix B. 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 w to which the rows $2i-1$ and $2i$ of B are assigned. The two paths leading from the root to v and w are identical up to some node u, where they split. By part 4 of Definition 1, 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. Furthermore, on each column corresponding to an edge on the paths from u to v and from u to w, 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 edges on the path between v and w. This path contains u. □

Given a genotype matrix A, we refer to the tree T_A with the labeling as described in Theorem 3 as a *directed perfect phylogeny* for A. An example of a genotype matrix and a directed perfect phylogeny for it is given in Fig. 1.

2.2. Perfect path phylogeny haplotyping

The variant of perfect phylogeny haplotyping that is central to this paper requires that the resulting perfect phylogeny takes the form of a path. Formally, a *perfect path phylogeny* is a perfect phylogeny consisting of at most two disjoint branches emanating from the root. It is convenient to consider a path phylogeny as partitioned into two *sides*, corresponding to the two branches (one of which may be empty). The problem of haplotyping via perfect path phylogenies is defined as follows:

Problem 4 (*Perfect Path Phylogeny Haplotyping,* {0, 1, 2}-PPPH).
Input: *A genotype* {0, 1, 2}*-matrix A*.
Question: *Does A admit a perfect path phylogeny*?

The motivation for considering path phylogenies in the context of haplotyping is the recent discovery that yin-yang (complementary) haplotypes are very common in human populations [22]. A genotype composed of a pair of yin-yang haplotypes results in an all-2 row in the genotype matrix, forcing any perfect phylogeny to take the form of a path.

Table 1
Statistics on the frequency of perfect path phylogenies in genotype data from Gabriel et al. [10].

Population	Window size	Genotype matrices	PPs (%)	PPPs (%)	RDH_{pp} (%)	RDH_{ppp} (%)
A	5	3029	51	40	16	11
	8	2843	29	19	3	1
	10	2720	20	13	< 1	< 1
B	5	2898	35	25	9	5
	8	2715	14	9	< 1	< 1
	10	2593	8	5	< 1	< 1
C	5	3000	59	51	21	15
	8	2817	37	28	5	5
	10	2695	27	20	2	1
D	5	2816	36	26	15	10
	8	2634	14	8	2	1
	10	2514	8	5	< 1	< 1

For each data set, genotype matrices were created by sliding a window of varying length over the original matrices. For the resulting matrices we list the percentage of those that admit a perfect phylogeny (PP) and a perfect path phylogeny (PPP). We also indicate what percentage of the genotype matrices satisfies the rich-data hypothesis while admitting a perfect phylogeny (RDH_{pp}) or perfect path phylogeny (RDH_{ppp})

To evaluate the abundance of perfect path phylogenies in real data, we considered genotype data sets from Gabriel et al. [10]. The data are from individuals grouped into four populations. For each individual, the genotypes were determined in 62 different regions of the genome. To assess the abundance of path phylogenies in these data sets, we examined consecutive windows of varying length in each genotype matrix. For a given window length l, we computed for every set of l consecutive columns in the input matrices: (a) whether these columns give rise to a perfect phylogeny; and (b) whether they give rise to a perfect path phylogeny. In the computation, we omitted rows that contained missing entries within the considered window. We checked whether these (complete) matrices admit a perfect phylogeny using the program PPH developed by Chung and Gusfield [4].

As shown in Table 1, approximately 70% of the instances that admit a perfect phylogeny also admit a perfect path phylogeny. Notably, only 23% of those genotype matrices admitting a perfect phylogeny satisfy the rich-data hypothesis and, thus, meet the requirements of the algorithm given by Halperin and Karp [16].

We end this section with a useful observation on perfect path phylogenies.

Lemma 5. *Let A be a matrix admitting a perfect path phylogeny T_A. If two columns c and d of A contain the submatrix $\begin{pmatrix} x & 0 \\ 0 & y \end{pmatrix}$ for some $x, y \in \{1, 2\}$, then they lie on different sides of T_A.*

Proof. Consider the path in T_A from the root to the edge labeled by c. By part 3 of Theorem 3, the columns that label edges on this path must attain a non-zero value in every row in which c attains a non-zero value. Hence, d cannot label an edge on this path. For the same reason, c cannot lie on the path from the root to d. □

2.3. Incomplete perfect phylogeny haplotyping

In practice, due to experimental noise, genotype data contain missing entries, manifested as question marks in the genotype matrix. A matrix with missing entries is called *incomplete*. An incomplete matrix *can be pp-completed* (*ppp-completed*) if the missing entries can be completed with values from {0, 1, 2} such that the resulting genotype matrix admits a perfect (path) phylogeny. The two arising problems are the incomplete perfect phylogeny haplotyping problem ({0, 1, 2, ?}-PPH) and the incomplete perfect path phylogeny haplotyping problem ({0, 1, 2, ?}-PPPH).

Problem 6 ({0, 1, 2, ?}-PPH).
Input: *A genotype* {0, 1, 2, ?}*-matrix A*.
Question: *Can A be pp-completed*?

Problem 7 ({0, 1, 2, ?}-PPPH).
Input: *A genotype* {0, 1, 2, ?}*-matrix A*.
Question: *Can A be ppp-completed*?

3. Hardness of haplotyping with missing data

In this section, we study the complexity of perfect phylogeny haplotyping with missing data. We show that already the special case of directed {0, 2, ?}-PPPH is NP-hard. In this version of the problem, the entries of the input matrix and its completions cannot attain the value 1. The hardness of this variant implies the NP-hardness of many other variants, see Theorem 9. In particular, perfect phylogeny haplotyping is NP-hard both in the undirected and the directed case, which was independently shown by Kimmel and Shamir [17].

Theorem 8. *Directed* {0, 2, ?}-PPPH *is* NP*-complete*.

Proof. Membership in NP is clear. The NP-hardness of {0, 2, ?}-PPPH is shown by reduction from NAE3SAT [11]. Given a Boolean formula in conjunctive normal form with three literals per clause, NAE3SAT calls for deciding whether there is an assignment to the variables such that in every clause at least one literal and at most two literals are satisfied.

Construction: Let ϕ 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 consists of three literals $\{l_1, l_2, l_3\}$, where each literal l_r is either a variable v_i or a negated variable $\bar{v}_i$. We map ϕ to a matrix A with entries from {0, 2, ?} with $2n + 3m$ rows and $2n + 3m$ columns. For $x \in \{0, 2, ?\}$ and a positive integer i, let x^i denote the all-x column vector of height i. For each variable v_i we define two column vectors:

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

For each clause $C_j = \{l_1, l_2, l_3\}$ we define a 3-column matrix:

$$\langle C_j \rangle := \begin{pmatrix} \langle l_1 \rangle & \langle l_2 \rangle & \langle l_3 \rangle \\ 0^{3(j-1)} & 0^{3(j-1)} & 0^{3(j-1)} \\ 2 & 0 & ? \\ 0 & ? & 2 \\ ? & 2 & 0 \\ 2^{3(m-j)} & 2^{3(m-j)} & 2^{3(m-j)} \end{pmatrix}.$$

The matrix A is composed of the following columns: For every variable v_i it contains the two *literal columns* $\begin{pmatrix} \langle v_i \rangle \\ 2^{3m} \end{pmatrix}$ and $\begin{pmatrix} \langle \bar{v}_i \rangle \\ 2^{3m} \end{pmatrix}$. For each clause C_j it contains the three *clause columns* of $\langle C_j \rangle$. The resulting matrix is illustrated in Fig. 2. We claim that ϕ is satisfiable iff A can be ppp-completed.

If-Part: Let A' be a completion of A that admits a directed perfect path phylogeny $T_{A'}$. We construct an assignment $\tau: \{v_1, \ldots, v_n\} \to \{0, 1\}$ from $T_{A'}$ as follows: choose any side of $T_{A'}$ and let $\tau(v_i) = 1$ iff the literal column $\begin{pmatrix} \langle v_i \rangle \\ 2^{3m} \end{pmatrix}$ labels an edge on this side. We make the following observations:

(1) By Lemma 5, the two columns of a variable must lie on different sides of $T_{A'}$, since they induce the submatrix $\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$.
(2) A clause column c corresponding to a literal l and the literal column corresponding to l lie on the same side of $T_{A'}$. To see that, observe that c and the literal column that corresponds to $\bar{l}$ induce the submatrix $\begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix}$ or the submatrix $\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$. Hence, Lemma 5 implies that they lie on different sides of $T_{A'}$. The previous observation shows that the two literal columns, corresponding to l and its negation, lie on different sides. Since $T_{A'}$ has at most two sides, the claim follows.

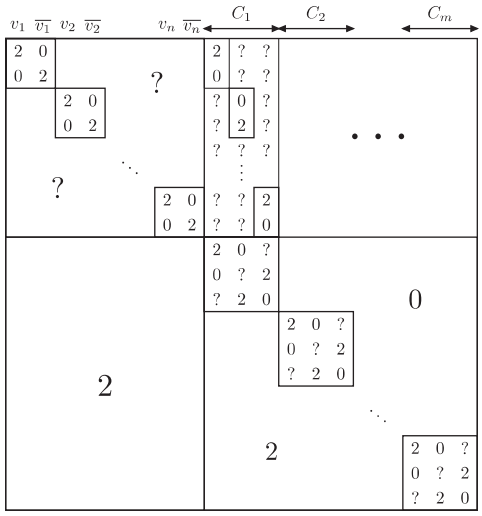

Fig. 2. Illustration of the matrix A from the proof of Theorem 8. It is constructed from a 3-CNF formula with variables $v_1, v_2, \ldots, v_n$ and clauses $C_1, C_2, \ldots, C_m$. In the above example, $C_1 = \{v_1, \bar{v}_2, v_n\}$.

(3) Each matrix $\langle C_j \rangle$ contains the submatrix $\begin{pmatrix} 2 & 0 & ? \\ 0 & ? & 2 \\ ? & 2 & 0 \end{pmatrix}$. Every completion of the missing entries induces one of the submatrices $\begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix}$ and $\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$. Hence, by Lemma 5, at most two edges out of the three edges labeled by these clause columns can lie on the same side of the perfect path phylogeny.

It follows that for any clause $C = \{l_1, l_2, l_3\}$ the literal and clause columns corresponding to these literals cannot all be on the same side of $T_{A'}$ and, thus, at most two literals of C are set to *true* and at most two are set to *false* by τ.

Only-If-Part: Let $\tau: \{v_1, \ldots, v_n\} \to \{false, true\}$ be a satisfying assignment. We extend it such that values are also assigned to the negated literals. We show how to derive a completion A' of A that admits a perfect path phylogeny. We start with the missing entries in the first $2n$ rows. We fill these entries in such a way that each of the completed rows equals one of the following two sequences of 0's and 2's: In the first sequence, the ith position is 2 if the literal corresponding to the ith column is assigned *true* by τ, and the ith position is 0 if the literal is assigned *false*. In the second sequence, the roles of 0- and 2-entries are exchanged. Note that for each of the first $2n$ rows the nonmissing entries in that row are, indeed, consistent with exactly one of these two sequences.

For the completion of the lower part of the clause columns consider a clause $C = \{l_1, l_2, l_3\}$. Exactly two of the literals are assigned the same truth value by τ. If l_i and l_j are these two literals, where $j = (i + 1) \bmod 33$, we replace the question mark in l_i's column by 2 and the question mark in l_j's column by 0. The question mark in the remaining column can be resolved arbitrarily, so we resolve it to, say, 0.

It remains to show that the resulting matrix A' has a perfect path phylogeny $T_{A'}$. We construct $T_{A'}$ as follows: all columns corresponding to literals that are assigned a value of 1 by τ are assigned to one side of $T_{A'}$, called the *true side*; all other literal columns are assigned to the other side of $T_{A'}$. Within each side, the edges labeled by these literal columns are ordered according to the literal indices, where a smaller index means that the edge is closer to the root of $T_{A'}$. Next, we add to each side of $T_{A'}$ edges labeled by clause columns. These edges are placed farther from the root than the literals' edges. Clause columns corresponding to literals that are assigned a value of *true* by τ are assigned to the true side; all other clause columns are assigned to the other side. The edges labeled by the clause columns are ordered according to the clause indices. Within each clause, if l_i and l_j are assigned to the same side and $j = (i + 1) \bmod 33$, l_i is put closer to the root than l_j.

We now show that the resulting path $T_{A'}$ satisfies the conditions of Theorem 3. Since A' does not contain 1-entries, it suffices to check that condition (3b) holds for every row of the matrix A'. The condition requires that the edges corresponding to 2-entries in each row form a path in $T_{A'}$ that visits the root. First, consider the two rows $2i - 1$ and $2i$ of A' that correspond to variable v_i. The columns in which row $2i - 1$ has value 2 form a complete path from the

root to one end of $T_{A'}$ (corresponding to literals with the same truth value), and the columns in which row $2i$ has value 2 form a complete path from the root to the other end of $T_{A'}$. Second, consider any three rows of A' in the lower part corresponding to a clause C_j, namely rows $2n+3j-2$, $2n+3j-1$, and $2n+3j$. For each of the three rows, the set of columns in which this row has value 2 forms a path in $T_{A'}$; this path extends between two clause columns that correspond to two literals of C_j that are assigned opposite truth values (including these columns or stopping just before them). □

The above theorem implies the hardness of more general variants of incomplete perfect phylogeny haplotyping:

Theorem 9. *Directed* {0, 1, 2, ?}-PPPH *and directed* {0, 1, 2, ?}-PPH *are* NP-*complete*.

Proof. Observe that directed {0, 1, 2, ?}-PPPH can be reduced to directed {0, 1, 2, ?}-PPH by adding an all-2 row to the {0, 1, 2, ?}-PPPH instance. By Theorem 3, this additional row ensures that any directed perfect phylogeny for the {0, 1, 2, ?}-PPH instance is necessarily a path. Hence, it suffices to establish the hardness of directed {0, 1, 2, ?}-PPPH. We reduce directed {0, 2, ?}-PPPH to directed {0, 1, 2, ?}-PPPH using the identity mapping: Let A be an input instance for {0, 2, ?}-PPPH. We map it to an instance $\hat{A} = A$ for the problem {0, 1, 2, ?}-PPPH. Clearly, a solution to A is also a solution to $\hat{A}$. Conversely, let $\hat{A}'$ be a completion of $\hat{A}$ that admits a perfect phylogeny. Let $T_{\hat{A}'}$ be a perfect phylogeny path for $\hat{A}'$. 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 phylogeny obtained from $T_{\hat{A}'}$ by replacing the column labels from $\hat{A}'$ with the corresponding columns in C. Then C is also a completion for A with only 0-entries and 2-entries. Since T_C satisfies the conditions of Theorem 3, C admits a perfect phylogeny. □

Since the directed variants of perfect phylogeny haplotyping can be reduced to the undirected ones by adding an all-0 row to the input matrix, we conclude:

Corollary 10. {0, 1, 2, ?}-PPPH *and* {0, 1, 2, ?}-PPH *are* NP-*complete*.

4. A partial-order perspective on haplotyping

This section provides the basic structural results for the design of our haplotyping algorithms. The main result of this section relates the existence of a perfect path phylogeny to properties of a partial order on the columns of the genotype matrix. The different partial orders studied in this section all require that a directed phylogeny is sought and we restrict attention to the directed case in the sequel. As mentioned earlier, for complete data one can reduce the undirected haplotyping problem to a directed one, see [6]; for incomplete data we show a result of this flavor in Section 6.1.

Let A be a genotype matrix and let T_A be a perfect phylogeny for A. Consider the following partial orders on the columns of A:

(1) The *ancestor relation* induced by the column labels of T_A. A column c is larger than another column c' with respect to this relation if c lies on the path from c' to the root.
(2) The *partial order* $\succcurlyeq$: Let $1 \succ 2 \succ 0$ and extend this order to {0, 1, 2}-columns by setting $c \succcurlyeq c'$ if $c[i] \succcurlyeq c'[i]$ for all rows i.
(3) The *leaf count order*: The *leaf count* of a column c is twice the number of 1-entries plus the number of 2-entries in c. This relation orders columns by increasing leaf count and considers columns as incomparable if they are different but have the same leaf count.

The first and the last order were introduced by Gusfield [15]; the order $\succcurlyeq$ was introduced by Eskin et al. [6], who implicitly showed that each order extends the one above it. Note that the last two relations exist even when there is no perfect phylogeny for A. In particular, they can be computed before the tree T_A is known. We note in passing that the set of all {0, 1, 2}-columns together with $\succcurlyeq$ is a graded lattice, where the leaf count provides the rank.

Let us review some basic definitions from poset theory. A *chain* is a linearly ordered subset of a poset. An *antichain* is a set of incomparable elements. The *width* of a poset is the size of its largest antichain. An element x *dominates* another element y if $x \succ y$.

The following theorem shows that the existence of a perfect path phylogeny for a matrix A with column set C can be decided based on properties of $(C, \succcurlyeq)$ alone.

Definition 11. Two columns are *separable* if each has a 0-entry in the rows where the other has a 1-entry. We say that a set C of $\{0, 1, 2\}$-columns has the *ppp-property* if it can be covered by two (possibly empty) chains $(C_1, \succcurlyeq)$ and $(C_2, \succcurlyeq)$, so that their maximal elements (if they exists) are separable. The pair (C_1, C_2) is called a *ppp-cover* of C.

Theorem 12. *A genotype matrix A admits a directed perfect path phylogeny iff its column set has the ppp-property.*

Proof. First, assume that A has a perfect path phylogeny T_A. If T_A has only one leaf, the claim holds, so assume T_A has two leaves. Let C_1 and C_2 be the sets of columns labeling the two paths from the root of T_A to each of its two leaves. Since $\succcurlyeq$ extends the ancestor relation in T_A, both $(C_1, \succcurlyeq)$ and $(C_2, \succcurlyeq)$ are chains. By part 3 of Theorem 3, for $i \in \{1, 2\}$, if a column $c \in C_i$ is maximal and has a 1-entry in some row r, then every column with a non-zero entry in this row must be in C_i. Consequently, all columns in the other chain must have a 0-entry in row r.

Conversely, assume that the column set C of A has the ppp-property and let (C_1, C_2) be a ppp-cover of C. For $i \in \{1, 2\}$ let p_i be a path whose edges are labeled by the columns in C_i. We construct T_A as follows: Let v_1 and v_2 be the end points of p_1 and p_2 that are incident to the edges labeled with the maximum elements of C_1 and C_2. We merge p_1 and p_2 by identifying v_1 with v_2, making the resulting vertex the root of T_A. Since the maximal elements of the two chains are separable, if a column in one chain has a 1-entry in row r, then all columns in the other chain have 0-entries in this row. Since the columns respect the $\succcurlyeq$ ordering, for $x, y \in \{0, 1, 2\}$ with $x \succ y$ a column with an x-entry in row r labels an edge that is closer to the root than a column along the same path with a y-entry in this row. Hence, the result is a perfect path phylogeny for A in the form of Theorem 3. □

We now introduce a compact representation of sets of columns with the ppp-property. It allows the ppp-property of a larger set to be checked given only the representation of the smaller set and the additional columns.

Let C be a set of $\{0, 1, 2\}$-columns. A maximal antichain in $(C, \succcurlyeq)$ is *highest of cardinality i* if it has cardinality i and no element of any other antichain of cardinality i dominates any of its elements. If a highest maximal antichain of cardinality i exists, it is unique and we denote it by $\mathrm{hma}_i(C)$. Otherwise, let $\mathrm{hma}_i(C)$ be the empty set. Let $\mathrm{hma}(C) := \mathrm{hma}_1(C) \cup \mathrm{hma}_2(C)$.

Theorem 13. *Let C and D be sets of columns so that no column in C has a larger leaf count than any column in D. Assume that C has the ppp-property. Then $C \cup D$ has the ppp-property if and only if* $\mathrm{hma}(C) \cup D$ *has it. Furthermore*, $\mathrm{hma}(C \cup D) = \mathrm{hma}(\mathrm{hma}(C) \cup D)$.

Proof. First, observe that the ppp-property is hereditary. In particular, if $C \cup D$ has it, then so does $\mathrm{hma}(C) \cup D$. Conversely, let (C_1, C_2) be a ppp-cover of C and let (D_1, D_2) be a ppp-cover of $\mathrm{hma}(C) \cup D$. We distinguish three cases:

(1) $\mathrm{hma}(C) = \{c\}$, where c is the maximal element of C. Since all elements of D have at least the leaf count of c, c is a minimal element of either D_1 or D_2. In the first case $(D_1 \cup C, D_2)$ is a ppp-cover of $C \cup D$, in the second case $(D_1, D_2 \cup C)$ is one.
(2) $\mathrm{hma}(C) = \{c_1, c_2\}$, where c_1 and c_2 are the (only) two maximal elements of C. Without loss of generality we may assume that c_1 is the maximal element of C_1, c_2 is the maximal element of C_2, $c_1 \in D_1$ and $c_2 \in D_2$. Since all elements in D have at least the leaf counts of c_1 and c_2, $(D_1 \cup C_1, D_2 \cup C_2)$ is a ppp-cover of $C \cup D$.
(3) $\mathrm{hma}(C) = \{c, c_1, c_2\}$, where $\mathrm{hma}_1(C) = \{c\}$ and $\mathrm{hma}_2(C) = \{c_1, c_2\}$. Without loss of generality we may assume $c_1 \in C_1 \cap D_1$ and $c_2 \in C_2 \cap D_2$. Let C_0 be the chain $\{c' \in C \mid c' \succ c_1\} = \{c' \in C \mid c' \succ c_2\}$. If $c \in D_1$, then $(D_1 \cup C_0 \cup C_1, D_2 \cup C_2 \backslash C_0)$ is a ppp-cover of $C \cup D$. If $c \in D_2$, then $(D_1 \cup C_1 \backslash C_0, D_2 \cup C_0 \cup C_2)$ is a ppp-cover of $C \cup D$.

Next, we show that $\mathrm{hma}(C \cup D) = \mathrm{hma}(\mathrm{hma}(C) \cup D)$. Denote $E := \mathrm{hma}(C) \cup D$. First, a maximum element of $C \cup D$ is also a maximum element of E and, thus, $\mathrm{hma}_1(C \cup D) = \mathrm{hma}_1(E)$. Let $H = \{h_1, h_2\} = \mathrm{hma}_2(C \cup D)$. Suppose to the contrary that $H \not\subseteq E$. The two elements h_1 and h_2 cannot be both in C or in D (or, else, H would have

been a subset of E), so without loss of generality $h_1 \in C \backslash \mathrm{hma}(C)$ and $h_2 \in D$. By definition, there exists an element $h \in \mathrm{hma}(C)$ that dominates h_1. Since the leaf count of h_2 is at least as high as that of h, they are incomparable. Hence, H is not the highest antichain of cardinality 2 in C, a contradiction.

It remains to consider the case that $H \subseteq E$. Suppose to the contrary that $\mathrm{hma}_2(E) \neq H$. Then one of the elements of $\mathrm{hma}_2(E)$ dominates h_1 or h_2 (or both). But then $\{h_1, h_2\}$ is not the highest maximal antichain of cardinality 2 in $C \cup D$, a contradiction. □

5. A linear-time algorithm for PPPH

In this section we present a linear-time algorithm for directed {0, 1, 2}-PPPH. The algorithm can be implemented efficiently and no large "hidden constants" are involved. In contrast, the fastest known algorithm for {0, 1, 2}-PPH is super-linear [15], and practical algorithms require quadratic time [1,6]. Our algorithm is based on a reduction to the problem of determining whether a given poset, together with a linear extension of it, has width at most 2. Felsner et al. [9] recently showed that this problem is solvable in linear time. The difference between their algorithm and the algorithm presented in the following is that instead of checking whether the poset $\succcurlyeq$ has width 2, we check whether it has the ppp-property. Recall that in addition to $\succcurlyeq$ having width 2, the ppp-property requires that the top elements of the ppp-cover are separable. The ideas used in the algorithm will also play a role in the more elaborate fixed-parameter algorithm presented in the next section.

The algorithm for PPPH: We order the columns $C = \{c_1, \ldots, c_m\}$ in order of ascending leaf counts. The algorithm constructs three stacks, denoted s, t_1, and t_2, having the following properties:

(1) The elements of each stack form a chain with respect to $\succcurlyeq$ with the largest element on top.
(2) All elements in s are larger than every element in t_1 or t_2.
(3) The top of stacks (*tos*) of t_1 and t_2 are incomparable.

In its main loop the algorithm iterates over the columns $c_1, \ldots, c_m$. For column c_i, the algorithm first checks whether this column and the tos of the non-empty stacks have the ppp-property. If this is the case, it checks whether c_i can be added on top of the stack s without violating the first two properties. If so, the algorithm pushes it onto s and proceeds with the next column. Otherwise, it checks whether c_i can be pushed onto t_1 or t_2 without violating the property that these stacks must be chains. If c_i cannot be added to either stack, no perfect path phylogeny exists since $\mathrm{tos}(t_1)$, $\mathrm{tos}(t_2)$, and c are three incomparable columns. Otherwise, c_i is pushed onto the stack admitting it, and stack s is moved on top of the other stack. For pseudo-code see Fig. 3.

Theorem 14. *Algorithm* PPPH *solves directed* {0, 1, 2}-PPPH *in linear time*.

Input: Genotype matrix A with column set C.
Output: "Yes" if A admit a perfect path phylogeny, "no" otherwise.

algorithm PPPH
  foreach $c \in C$ **do**
    compute the leaf count of column c
  sort C according to leaf counts using bucket sort
  $s \leftarrow$ empty stack, $t_1 \leftarrow$ empty stack, $t_2 \leftarrow$ empty stack
  foreach $c \in C$ in order of ascending leaf count **do**
    if $\{\mathrm{tos}(s), \mathrm{tos}(t_1), \mathrm{tos}(t_2), c\}$ does not have the ppp-property **then**
      output "no"
    if $c \succeq \mathrm{tos}(s)$ and $c \succeq \mathrm{tos}(t_1)$ and $c \succeq \mathrm{tos}(t_2)$ **then push** c onto s
    else if $c \succeq \mathrm{tos}(t_1)$ **then push** c onto t_1; **move** all of s on top of t_2
    else if $c \succeq \mathrm{tos}(t_2)$ **then push** c onto t_2; **move** all of s on top of t_1
    else output "no"
  output "yes"

Fig. 3. A linear-time algorithm for computing a perfect path phylogeny. In the algorithm, if a stack is empty, a condition like $c \succcurlyeq \mathrm{tos}(s)$ is considered to be true.

Proof. *Correctness*: Whenever the algorithm outputs "no", it has detected a set of columns in the genotype matrix that violates the ppp-property. It remains to show that on output "yes" a perfect path phylogeny exists.

Let $C = \{c_1, \ldots, c_m\}$ be the set of columns of A in order of ascending leaf counts. After column c_i has been processed in iteration i, let $S_i := \{\text{tos}(s)\}$ (let S_i be empty if s is empty) and $T_i := \{\text{tos}(t_1), \text{tos}(t_2)\}$. We prove by induction that

$$S_i = \text{hma}_1(\{c_1, \ldots, c_i\})$$

and

$$T_i = \text{hma}_2(\{c_1, \ldots, c_i\}).$$

For $i = 0$ we have $S_0 = T_0 = \emptyset$ and the claim is trivially true. Assume that the claims hold for $i-1$ and we add the column c_i. If the first push-statement is executed, $S_i = \{c_i\}$. Since c_i dominates every other element, $S_i = \text{hma}_1(\{c_1, \ldots, c_i\})$. Furthermore, we have $T_i = T_{i-1}$ and $\text{hma}_2(\{c_1, \ldots, c_i\}) = \text{hma}_2(\{c_1, \ldots, c_{i-1}\})$.

If either the second or third push-statements are executed, the set T_i contains two incomparable elements and no element dominates either of them. This shows $\text{hma}_1(\{c_1, \ldots, c_i\}) = \emptyset$ and $\text{hma}_2(\{c_1, \ldots, c_i\}) = T_i$. Since the stack s is emptied, we have $S_i = \emptyset$.

Putting it all together, we conclude that at the end of each iteration of the algorithm $T_i \cup S_i = \text{hma}(\{c_1, \ldots, c_i\})$. As we also check every time whether $\{c_i\} \cup T_{i-1} \cup S_{i-1}$ has the ppp-property, Theorem 13 tells us that after each iteration $\{c_1, \ldots, c_i\}$ has the ppp-property. Thus, the final output "yes" is correct.

Running time: We claim that the algorithm runs in $O(mn)$ time for an $n \times m$ matrix. Computing the leaf count of a column takes time $O(n)$, so computing all leaf counts takes time $O(mn)$. For the sorting we use bucket sort. There are at most $2n$ buckets since the leaf counts range from 0 to $2n$. Thus the sorting can also be done in time $O(mn)$. The main loop has at most m iterations, each requiring $O(n)$ time. Overall, the algorithm runs in time $O(mn)$. □

6. A fixed-parameter algorithm for incomplete PPPH

In this section we develop a fixed-parameter algorithm for {0, 1, 2, ?}-PPPH, where the parameter is the maximum number of missing entries in a column. The section is organized as follows: In Section 6.1 we examine the characteristics of real genotype data sets. We show that one can restrict attention to the directed case for virtually all considered sets and that parameterizing the maximum number of missing entries in a column is adequate given the distribution of missing entries in real data. In Section 6.2 we give an overview of the structure of the fixed-parameter algorithm. The algorithm consists of two phases, a preprocessing phase and a dynamic programming phase, whose descriptions are given in Sections 6.3 and 6.4, respectively.

6.1. Characteristics of genotype matrices

Eskin et al. have given a reduction from undirected to directed perfect phylogeny haplotyping on complete data [6]. However, for incomplete matrices no general reduction to the directed case is known. Below we describe an extension of the reduction in [6] to incomplete matrices. Although this extension cannot be applied in all cases, it works for virtually all real data sets we examined.

We say that a genotype matrix A can be *directed* if one can transform A into a genotype matrix $\tilde{A}$ such that A admits an undirected perfect phylogeny if and only if $\tilde{A}$ admits a directed perfect phylogeny. Key to the reduction in [6] is the following proposition:

Proposition 15. *Let A be a genotype matrix. For every pair of columns in A let there exist a row with either two* 0*-entries or one* 0*-entry and one* 2*-entry in these columns. Then A admits a perfect phylogeny if and only if A admits a directed perfect phylogeny.*

Using this proposition, one proceeds as follows to direct a genotype matrix A: every column is relabeled such that when going down the column, its first entry with a value different from 2 is a 0-entry. If this is not the case, we exchange the 0- and 1-entries within that column.

We can extend this transformation to genotype matrices with missing data by trying to permute the rows of A such that within each column there are no missing entries up to the first 0- or 1-entry. If such a permutation of rows exists,

Table 2
Statistics on the distribution of ?-entries in genotype data from [10]

Population	A	B	C	D
Number of individuals (rows)	93	50	42	96
Number of genotype matrices	62	62	62	62
Average maximum number of ?'s per column	21.7	10.5	8.3	21.3
Average number of ?'s per column	6.2	2.7	1.6	6.8
Percentage of genotype matrices that can be directed	97%	100%	100%	100%

it can be found greedily and we can proceed as in the original reduction of Eskin et al. [6] to produce a matrix $\tilde{A}$ for which we can search for a directed perfect phylogeny.

The above reduction to the directed case depends on the existence of an appropriate row permutation. To check how often such a permutation exists, we examined an extensive collection of real data sets from [10]. As shown in Table 2, virtually all data sets that we checked can be directed (246 out of 248). Thus, the data suggest that we can focus on solving the haplotyping problem in the directed case.

In addition, we tested the assumptions made on the distribution of missing entries by evaluating what portion of the data is missing and whether the missing data entries are distributed in a random way. The results are summarized in Table 2. One observes that the maximum number of missing entries per column is approximately one fifth of the total column size, and the average number of missing entries per column is small.

6.2. Structure of the algorithm

The input to the algorithm is a $\{0, 1, 2, ?\}$-matrix A of dimension $n \times m$ with at most k missing entries per column. The algorithm proceeds in two phases. In the first phase, which we call the *preprocessing phase*, the input is simplified by collapsing multiple columns to one consensus column under certain conditions, reducing the size of the matrix. In the second phase, dynamic programming is used to compute a completion of the missing entries in a way admitting a perfect path phylogeny.

The core idea of the preprocessing phase is the following: suppose several columns become identical when some ?-entries are completed appropriately, and suppose we replace them with that one "consensus" column. Clearly, if we can find a perfect path phylogeny for this new matrix, we can also find one for the original matrix. The more difficult observation is that the reverse implication is also true if the number of columns that formed the consensus is large enough.

The dynamic program iterates over the columns of the preprocessed input matrix in order of increasing leaf count, building perfect path phylogenies from the leaves toward the root. The preprocessing ensures that for each leaf count there is only a fixed number of columns with that leaf count. For each column we consider all possible ways to complete the ?-entries in it. For each possible completion we check whether a perfect path phylogeny exists and, if so, record information about it. The crucial observation, which makes the dynamic programming feasible, is that the perfect path phylogenies for columns up to a certain leaf count can be constructed from the information we stored for columns of a constant-size range of leaf counts preceding the current leaf count.

6.3. Preprocessing phase

The objective of the preprocessing is to ensure that there will only be a fixed number of columns that have any given leaf count. For a complete matrix, three or more distinct columns with the same leaf count imply that the matrix does not admit a perfect path phylogeny due to Lemma 5.

In the presence of missing data there can be an arbitrary number of distinct columns that have the same leaf count (just like 0-entries, ?-entries do not count), but still the genotype matrix allows a perfect path phylogeny. The reason is that we may be able to "collapse" all the different columns into one column by filling up the missing entries appropriately. However, we cannot just collapse everything that can be collapsed, as there are cases in which it is necessary *not* to

Input: Genotype matrix A with at most k missing entries per column.
Output: A preprocessed matrix, in which every column has dimension at most $\kappa(k)$ and which can be ppp-completed if and only if A can.

```
procedure preprocess
    repeat
        for l ← 1 to k do
            foreach column c of A do
                foreach partial completion p of c do
                    if p has k − l many ?-entries then
                        C ← {c' | c' is a column of A and
                                  a partial completion of p}
                        if |C| > κ(l) then
                            A ← A with all columns in C replaced by p
                            break for loop
    until A remains unchanged
```

Fig. 4. The preprocessing procedure.

collapse columns in order to obtain a perfect path phylogeny. For example, consider the following matrix:

$$\begin{pmatrix} ? & 0 & 0 & 2 & 0 \\ 0 & ? & 0 & 0 & 2 \\ 2 & 2 & ? & 0 & 0 \end{pmatrix}.$$

The first three columns can be collapsed to the single column $\begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}$, but the resulting matrix $\begin{pmatrix} 0 & 2 & 0 \\ 0 & 0 & 2 \\ 2 & 0 & 0 \end{pmatrix}$ does not admit a perfect path phylogeny. However, completing the ?-entries in the first two columns to 2 and in the third column to 0, yields a matrix that admits a perfect path phylogeny.

In the following we identify situations in which it is safe to collapse columns.

Terminology: A (*partial*) *completion* of a column c with ?-entries is a column c' obtained by replacing (some of) the ?-entries with 0, 1, or 2. A *completion of a set* C of columns with ?-entries is a set containing one completion of each $c \in C$. Let completions (C) denote the set of all completions of a set C of columns. For convenience, we define completions$(\emptyset) := \{\emptyset\}$. Since every column has at most k ?-entries, every column has at most 3^k completions. Thus, $|\text{completions}(C)| \leqslant 3^{k|C|}$.

A *consensus* for a set C of columns is a column c that is a partial completion of all $c' \in C$. The columns in C are said to be *consenting to* c. Note that two columns c and c' are consenting if and only if $c[i] \neq c'[i]$ implies $c[i]$=? or $c'[i]$=? for all rows i. A consensus can contain ?-entries only in rows in which all columns in C have a ?-entry. For an incomplete genotype matrix A and a column c (not necessarily in A), the *dimension* of c in A is the size of largest subset C of columns of A such that c is a consensus of C.

The preprocessing procedure: Define $\kappa(l) := 6^l \cdot l!$. The preprocessing algorithm repeatedly does the following: for $l = 1, \ldots, k$, it scans the input matrix for a collapsible set of columns of size at least $\kappa(l)$, and collapses those. If no such set is found, the algorithm stops. We find the collapsible sets as follows: first, we compute for each column c of A all possible partial completions. Since each column admits at most 4^k distinct partial completions, there are at most $4^k m$ "candidates" for a consensus. Let P_l be the set of all candidates that have exactly $k - l$ question mark entries. We can then check for each $p \in P_l$ whether there are more than $\kappa(l)$ different columns in A that have p as consensus. For pseudo-code see Fig. 4.

To prove the correctness of the algorithm, we use the following two lemmas.

Lemma 16. *For* $x, y_1, y_2, y_3 \in \{0, 1, 2\}$ *and* $x \notin \{y_1, y_2, y_3\}$, *the matrix* $C = \begin{pmatrix} y_1 & x & x \\ x & y_2 & x \\ x & x & y_3 \end{pmatrix}$ *does not admit a perfect path phylogeny.*

Proof. For $1 \leqslant i < j \leqslant 3$ consider the submatrix $\begin{pmatrix} y_i & x \\ x & y_j \end{pmatrix}$ and consider a resolution of this genotype submatrix into haplotypes. This resolution must contain the identity submatrix. By Lemma 5 the columns i and j cannot be placed on the same side in a perfect path phylogeny. □

Lemma 17. *Let A be a genotype matrix with at most k missing entries per column. Let C be a set of columns in A and let $l \geqslant 1$ be minimal such that the columns in C have a consensus c containing $k - l$ missing entries. If $|C| > \kappa(l)$, then the matrix obtained from A by replacing all columns in C by c can be ppp-completed if and only if A can be ppp-completed.*

Proof. Clearly, if the collapsed matrix can be ppp-completed, so can A. Suppose to the contrary that A has a completion A' that admits a perfect path phylogeny, but the collapsed matrix cannot be ppp-completed. Let C' be the set of columns in A' that complete the columns in C. There can be no column $d' \in C'$ that is consenting with c since, otherwise, d' would complete all columns in C and, therefore, the collapsed matrix could be ppp-completed.

We prove our claim by induction on l. For $l = 1$, our assumption is that $|C| > 6$. The consensus c contains $k - 1$ question mark entries. Thus, each column $d \in C$ can contain at most one additional ?-entry. Since d is consenting with c but the corresponding completed column $d' \in C'$ is not, we conclude that d' and c disagree in the position at which d has its additional question mark. We call this the *disputed* position of d.

We extract those rows in C, C', and c that contain the disputed positions of columns in C. Since no two columns in C are identical, every disputed ?-entries in C must be in a different row. Thus, we extract at least seven rows, at least three of which must have the same value in the consensus. We extract such three rows from C along with three columns that have a ?-entry in these rows. The resulting submatrix is of the form: $\begin{pmatrix} ? & x & x \\ x & ? & x \\ x & x & ? \end{pmatrix}$, where $x \in \{0, 1, 2\}$. Since the columns disagree with the consensus c, the ?-entries cannot be completed to x. By Lemma 16, for all $x \in \{0, 1, 2\}$ both this matrix and hence also A itself cannot be ppp-completed, a contradiction.

For the inductive step, suppose that we have already established the claim for $l - 1$ (and all k). Now, suppose a consensus c with $k - l$ question mark entries exists that has dimension larger than $\kappa(l)$, but that every column with $k - l'$ question mark entries has dimension at most $\kappa(l')$ for all $l' < l$. Consider those rows in C that contain a ?-entry, while the corresponding position in c contains no question mark. The submatrix of C obtained in this way has the following property: Each row contains at most $\kappa(l - 1)$ missing entries. Indeed, if this were not the case, then the columns of C containing these ?-entries would have a consensus $\tilde{c}$ having at least one more ?-entry than c does and dimension more than $\kappa(l - 1)$. This would contradict the minimality of l.

For each column $d \in C$, there is at least one ?-entry that faces a non-question mark entry in c and that is completed differently in C' from the entry in c. As above, let us call these positions the *disputed* positions of d.

We form sets C_x for $x \in \{0, 1, 2\}$ containing those columns in C that have a disputed position facing entry x in the consensus c. Since every column in C has a disputed position, we have $C = C_0 \cup C_1 \cup C_2$. We claim that each of the sets C_0, C_1, and C_2 has size at most $2l \cdot \kappa(l - 1)$, which would imply that C has size at most $6l \cdot \kappa(l - 1)$, a contradiction.

Let us start with C_0. We construct a graph G_0 whose vertex set is exactly C_0. Let there be an edge between a vertex $d \in C_0$ and $e \in C_0$ if there is a row in which both d and e have a disputed position that faces a 0-entry in c. We claim that the maximum degree of G_0 is less than $l\kappa(l - 1)$. To see this, first note that every vertex (which is a column) has at most l disputed positions. Next, we already saw that in each row there are at most $\kappa(l - 1)$ many question mark entries. Thus, each vertex can be adjacent to at most $l \cdot (\kappa(l - 1) - 1)$ different vertices.

We claim that the largest independent set in G_0 has size at most 2. To see this, assume that three independent columns are given and consider those rows where these columns have a disputed position. Stripped of everything but these three columns and rows, the matrix is once more $\begin{pmatrix} ? & 0 & 0 \\ 0 & ? & 0 \\ 0 & 0 & ? \end{pmatrix}$. Since all the question marks are at disputed positions, we know that in the matrix A' they are all completed differently from 0. By Lemma 16, this is not possible.

We have just shown that the graph G_0 has maximum degree $l \cdot (\kappa(l - 1) - 1)$ and independence number at most 2. This implies that it can have at most $2l \cdot (\kappa(l - 1) - 1) + 2 < 2l \cdot \kappa(l - 1)$ vertices, where the inequality follows from the assumption $l > 1$. Therefore, $|C_0| < 2l \cdot \kappa(l - 1)$. Using the same argument we can prove the same bounds for C_1 and C_2. □

Theorem 18. *Algorithm preprocess is a fixed-parameter algorithm that transforms a genotype matrix A into a genotype matrix $\bar{A}$ such that $\bar{A}$ can be ppp-completed if and only if A can. Furthermore, if more than $\lambda(k) := (4k + 2)\kappa(k)$ columns in $\bar{A}$ have the same leaf count, then neither A nor $\bar{A}$ can be ppp-completed.*

Proof. *Correctness*: By Lemma 17, whenever columns are collapsed in the algorithm, the resulting matrix can be ppp-completed if any only if the A can. Consider the matrix $\bar{A}$ that is output at the end of the algorithm. For a leaf

count i, consider the set D of columns in $\bar{A}$ having leaf count i. Suppose $|D| > (4k+2)\kappa(k)$. Since no more than $\kappa(k)$ columns in $\bar{A}$ have a consensus, in any completion of $\bar{A}$ the columns in D must be completed in at least $4k+3$ different ways. On the other hand, each completion of a column from D must have a leaf count between i and $i+2k$ and a perfect path phylogeny can contain at most two columns with the same leaf count. Thus, there are at most $2(2k+1) = 4k+2$ different ways of completing columns in D; a contradiction. This shows that $\bar{A}$ cannot be ppp-completed if more than $(4k+2)\kappa(k)$ columns have the same leaf count.

Running time: In each iteration of the main repeat-loop the number of columns in the genotype matrix decreases by at least 1 since some columns are collapsed. Thus there can be at most m iterations of the main loop. The two for each-statements iterate over at most $m4^k$ candidates and computing each candidate takes time at most $O(n)$. Computing the set C from a given column p can be done in time $O(mn)$. Putting it all together, the algorithm terminates after at most $O(k4^k m^3 n)$ steps. □

6.4. Dynamic programming phase

The input for the dynamic programming phase is a preprocessed $n \times m$ matrix A. Observe that the leaf counts of the columns of A range between 0 and some number $L \leqslant 2n$. We use $A_{[i,i']}$ with $0 \leqslant i \leqslant i' \leqslant L$ to denote set of columns of A that have leaf counts between i and i'. The set $A_{[i-2k,i]}$ contains all columns of A whose leaf count could become i after completion of the missing entries. For a set C of {0, 1, 2}-columns, we use $C^{=j}$ to denote the columns in C that have leaf count j; $C^{\leqslant j}$ and $C^{\geqslant j}$ are similarly defined. Thus, we use subscripts to refer to the leaf count *before* replacing question marks, and superscripts to refer to the leaf count *after* completion. We say that two sets of completed columns are *consistent* if, whenever both sets contain completions of the same original column, these completions are identical.

The dynamic programming algorithm fills a table $\mathscr{H}(R, i)$, where the "row" R indexes sets of completed columns of A, and the "column" i indexes leaf counts. A column i will be processed only if some column of A can be completed so that its leaf count becomes i; otherwise it will be skipped during a run of the algorithm. Column i has a row entry for every $R \in \text{completions}(A_{[i-2k,i]})$. Informally, an entry $\mathscr{H}(R, i)$ stores information about all completions of columns in $A_{[0,i]}$ that are consistent with R and that allow a perfect path phylogeny. Formally,

$$\begin{aligned}\mathscr{H}(R,i) := \{\text{hma}(C^{\leqslant i}) \mid\ & C \in \text{completions}(A_{[0,i]}),\\ & C \text{ is consistent with } R,\\ & C \text{ has the ppp-property}\}.\end{aligned} \tag{1}$$

As we show in the following lemma, the information recorded in each entry of the table suffices for extending the perfect path phylogeny in the subsequent dynamic programming step:

Lemma 19. *Let* $i \in \{1, \ldots, L\}$ *and* $R \in \text{completions}(A_{[i-2k,i]})$. *Then*

$$\begin{aligned}\mathscr{H}(R,i) = \{\text{hma}(R^{=i} \cup H) \mid\ & S \in \text{completions}(A_{[i-1-2k,i-1]}),\\ & S \text{ is consistent with } R,\\ & H \in \mathscr{H}(S, i-1),\\ & R^{\geqslant i} \cup H \text{ has the ppp-property}\}.\end{aligned} \tag{2}$$

Proof. *Left-to-right-inclusion*: Consider any $\text{hma}(C^{\leqslant i}) \in \mathscr{H}(R, i)$, where by definition $C \in \text{completions}(A_{[0,i]})$, C is consistent with R, and C has the ppp-property. Since C is consistent with R, we must have $R \subseteq C$. In particular, $C^{=i} = R^{=i}$. Since $C^{\leqslant i} = C^{=i} \cup C^{\leqslant i-1}$, Theorem 13 implies that

$$\text{hma}(C^{\leqslant i}) = \text{hma}(C^{=i} \cup C^{\leqslant i-1}) = \text{hma}(C^{=i} \cup \text{hma}(C^{\leqslant i-1})).$$

Set $H := \text{hma}(C^{\leqslant i-1})$. Observe that $R^{\geqslant i} \cup H$ has the ppp-property since its superset C has it. Hence, it suffices to prove the existence of a set of columns $S \in \text{completions}(A_{[i-1-2k,i-1]})$, such that S is consistent with R and $H \in \mathscr{H}(S, i-1)$. Let S be the set of columns from C that had leaf counts in the range $[i-1-2k, i-1]$ *before* their completion. Since C is consistent with R, so is S. Furthermore, $H \in \mathscr{H}(S, i-1)$ since the set of all columns in C that had leaf count in the range $[0, i-1]$ before their completion is in $\text{completions}(A_{[0,i-1]})$, is consistent with S, and inherits the ppp-property from C.

Input: Genotype matrix A with at most k missing entries per column.
Output: "Yes" if A can be ppp-completed, "no" otherwise.

procedure *dynamic_program_init*
  foreach $R \in \text{completions}(A_{[0,0]})$
    if R has the ppp-property **then** $\mathcal{H}(R,0) \leftarrow \{\text{hma}(R^{=0})\}$
      else $\mathcal{H}(R,0) \leftarrow \emptyset$

procedure *dynamic_program_step*(i)
  foreach $R \in \text{completions}(A_{[i-2k,i]})$
    $\mathcal{H}(R,i) \leftarrow \emptyset$
    foreach $S \in \text{completions}(A_{[i-1-2k,i-1]})$
      if R and S are consistent **then**
        $\mathcal{H}(R,i) \leftarrow \mathcal{H}(R,i) \cup \{\text{hma}(R^{=i} \cup H) \mid H \in \mathcal{H}(S,i-1), R^{\geqslant i} \cup H \text{ has the ppp-property}\}$

algorithm INCOMPLETE-PPPH
  call *preprocess*
  for $i \leftarrow 0$ **to** L **do**
    if $|A_{[i,i]}| > \lambda(k)$ **then output** "no"; **stop**
  call *dynamic_program_init*
  for $i \leftarrow 1$ **to** L **do**
    call *dynamic_program_step*(i)
  if $\bigcup_{R \in A_{[L-2k,L]}} \mathcal{H}(R,L) \neq \emptyset$ **then output** "yes"
    else output "no"

Fig. 5. The fixed-parameter algorithm for directed {0, 1, 2, ?}-PPPH.

Right-to-left-inclusion: Let $S \in \text{completions}(A_{[i-1-2k,i-1]})$ be given and $H \in \mathcal{H}(S, i-1)$ such that S is consistent with R and $H \cup R^{\geqslant i}$ has the ppp-property. By definition $H = \text{hma}(D^{\leqslant i-1})$ for some $D \in \text{completions}(A_{[0,i-1]})$ that is consistent with S and has the ppp-property. Note that D is also consistent with R and, thus, we can decompose $D \cup R = (D \cup R)^{\leqslant i-1} \cup (D \cup R)^{\geqslant i} = D^{\leqslant i-1} \cup R^{\geqslant i}$. Since $H \cup R^{\geqslant i}$ has the ppp-property, Theorem 13 implies that $D^{\leqslant i-1} \cup R^{\geqslant i}$ also has it. Let $C := D^{\leqslant i-1} \cup R^{\geqslant i} = D \cup R$. Then C is consistent with R, it has the ppp-property, and $C \in \text{completions}(A_{[0,i]})$. Thus, $\text{hma}(C^{\leqslant i}) \in \mathcal{H}(R,i)$. Since $C^{\leqslant i} = R^{=i} \cup D^{\leqslant i-1}$, we conclude that $\text{hma}(C^{\leqslant i}) = \text{hma}(R^{=i} \cup \text{hma}(D^{\leqslant i-1})) = \text{hma}(R^{=i} \cup H)$. □

The algorithm for incomplete PPPH: The dynamic programming algorithm starts with a preprocessed matrix A. The algorithm's objective is to fill the dynamic programming table $\mathcal{H}$ column-wise. First, we fill column 0. For each $R \in \text{completions}(A_{[0,0]})$, we store $\{\text{hma}(R^{=0})\}$ in $\mathcal{H}(R,0)$ if R has the ppp-property; otherwise the table entry is set to $\emptyset$. Note that if $\mathcal{H}(R,0) \neq \emptyset$, then either $\mathcal{H}(R,0) = \{\emptyset\}$ (when $R^{=0} = \emptyset$) or $\mathcal{H}(R,0)$ is exactly the set containing the all-0 column.

For the dynamic programming step, assume that we have filled column $i-1$ and now wish to fill column i. We compute the entries $\mathcal{H}(R,i)$ using Eq. 2. We iterate over all sets $S \in A_{[i-1-2k,i-1]}$ that are consistent with R. For each such set S we iterate over all the elements $H \in \mathcal{H}(S, i-1)$. If $R^{\geqslant i} \cup H$ has the ppp-property, we add $\text{hma}(H \cup R^{=i})$ to $\mathcal{H}(R,i)$. At the end, we check whether the last column L contains some non-empty entry. For pseudo-code see Fig. 5.

Note that the algorithm only determines whether the matrix admits a perfect path phylogeny, but can be easily modified to output a perfect path phylogeny if such exists.

Theorem 20. *Algorithm* INCOMPLETE-PPPH *is a fixed-parameter algorithm that solves directed* {0, 1, 2, ?}-PPPH, *where the parameter is the maximum number of missing entries in a column.*

Proof. *Correctness*: Eq. (1) trivially holds for the first column of the dynamic programming table. By Lemma 19, we correctly fill all other columns of $\mathcal{H}$. Thus, on the one hand, after the algorithm has terminated every non-empty entry in the last column of $\mathcal{H}$ corresponds, by Eq. (1), to a completion A' of the input matrix that admits a perfect path phylogeny. On the other hand, every ppp-completion A' of A is in particular a completion of the columns in $A_{[L-2k,L]}$.

Hence, by Eq. (1), the entry of the dynamic programming table that corresponds to this completion is non-empty, since it contains $\mathrm{hma}(A'^{\leqslant L})$.

Running time: We start with a loose bound on the size of the entries $\mathscr{H}(R, i)$. Every entry is a list of hma-sets, which are sets containing at most three completed columns. There are at most $3^{3k}m^3$ ways to choose the hma-sets. This shows that $|\mathscr{H}(R, i)| \leqslant 3^{3k}m^3$. (With a more elaborate argument one can show a bound that is linear in m.)

The algorithm iterates over columns of the table $\mathscr{H}$, whose number is at most $L \leqslant 2n$. For each column two nested for each-loops iterate over the elements of the sets $\mathrm{completions}(A_{[i-2k,i]})$ and $\mathrm{completions}(A_{[i-1-2k,i-1]})$. By Theorem 18, $|A_{[i-2k,i]}| \leqslant (2k+1) \cdot \lambda(k)$ and, thus, $|\mathrm{completions}(A_{[i-2k,i]})| \leqslant 3^{k(2k+1)\cdot\lambda(k)}$. Finally, in each update of $\mathscr{H}(R, i)$ the algorithm iterates over all elements of a list with at most $3^{3k}m^3$ elements and performs a check that takes time $\mathrm{O}(n)$. In total, since the preprocessing can be done in $\mathrm{O}(4^k m^3 n)$, the total running time of the algorithm is bounded by $3^{\mathrm{O}(k^2\cdot 6^k\cdot k!)}\, n^2 m^3$. □

7. Conclusions

In this paper, we have introduced the concept of perfect path phylogenies and demonstrated how it can be applied to haplotyping with missing data. While the general problem of incomplete perfect phylogeny haplotyping was shown to be NP-hard even in very restricted cases, we have given a fixed-parameter algorithm for the problem when the sought phylogeny is a path. We have also shown that the majority (70%) of real data sets that admit a perfect phylogeny also admit a path phylogeny.

Our work is a step toward coping with the abundance of missing entries in genotype data. Our algorithms exploit connections between a perfect phylogeny for genotype matrices and a partial order on its columns. Instead of path phylogenies one could study phylogenies whose underlying posets have bounded width. It would be interesting to generalize our methods to this case and to check how well real data conforms to this relaxed restriction.

We remark that there exist different results that relate the complexity of perfect phylogeny haplotyping problems to classical problems on hypergraph realization, see [2,15]. Our proof of the NP-hardness of {0, 2, ?}-PPPH implies the NP-hardness of certain sandwich problems arising in hypergraph realization, see [12] for details. In particular, our results imply that the sandwich problem for hypergraph tree realization is NP-hard.

Two remaining open questions are whether the fixed-parameter tractability of the incomplete perfect path phylogeny haplotyping with respect to the number of missing entries can be maintained for the undirected case and, more importantly, for general (non-path) phylogenies.

Acknowledgments

We would like to thank Eran Halperin for insightful discussions on haplotyping with missing data and Arfst Nickelsen for suggesting improvements of the manuscript.

References

[1] V. Bafna, D. Gusfield, G. Lancia, S. Yooseph, Haplotyping as perfect phylogeny: a direct approach, J. Comput. Biol. 10 (3–4) (2003) 323–340.
[2] T. Barzuza, J.S. Beckmann, R. Shamir, I. Pe'er, Computational problems in perfect phylogeny haplotyping: XOR-genotypes and tag SNPs, in: Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM), Lecture Notes in Computer Science, vol. 3109, Springer, Berlin, 2004, pp. 14–31.
[3] R.H. Chung, D. Gusfield, Empirical exploration of perfect phylogeny haplotyping and haplotypers, in: Proceedings of the Ninth International Conference on Computing and Combinatorics, Lecture Notes in Computer Science, vol. 2697, Springer, 2003, pp. 5–19.
[4] R.H. Chung, D. Gusfield, Perfect phylogeny haplotyper: haplotype inferral using a tree model, Bioinformatics 19 (6) (2003) 780–781.
[5] A. Clark, Inference of haplotypes from PCR-amplified samples of diploid populations, J. Mol. Biol. Evol. 7 (2) (1990) 111–122.
[6] E. Eskin, E. Halperin, R.M. Karp, Efficient reconstruction of haplotype structure via perfect phylogeny, J. Bioinformatics Comput. Biol. 1 (1) (2003) 1–20.
[7] E. Eskin, E. Halperin, R. Sharan, Optimally phasing long genomic regions using local haplotype predictions, in: Proceedings of Second RECOMB Satellite Workshop on Computational Methods for SNPs and Haplotypes, Pittsburgh, Pennsylvania, 2004, pp. 13–26.
[8] L. Excoffier, M. Slatkin, Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population, Mol. Biol. Evol. 12 (5) (1995) 921–927.
[9] S. Felsner, V. Raghavan, J. Spinrad, Recognition algorithms for orders of small width and graphs of small Dilworth number, Order 20 (2003) 351–364.

[10] S.B. Gabriel, S.F. Schaffner, H. Nguyen, J.M. Moore, J. Roy, B. Blumenstiel, J. Higgins, M. DeFelice, A. Lochner, M. Faggart, S.N. Liu-Cordero, C. Rotimi, A. Adeyemo, R. Cooper, R. Ward, E.S. Lander, M.J. Daly, D. Altshuler, Structure of halpotype blocks in the human genome, Science 296 (2002) 2225–2229.
[11] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, San Francisco, 1979.
[12] J. Gramm, T. Nierhoff, R. Sharan, T. Tantau, On the complexity of haplotyping via perfect phylogeny, in: Proceedings of Second RECOMB Satellite Workshop on Computational Methods for SNPs and Haplotypes, Pittsburgh, Pennsylvania, 2004, pp. 35–46.
[13] J. Gramm, T. Nierhoff, T. Tantau, Perfect path phylogeny haplotyping with missing data is fixed-parameter tractable, in: Proceedings of the 2004 International Workshop on Parameterized and Exact Computation, Lecture Notes in Computer Science, vol. 3162, Springer, Berlin, 2004, pp. 174–186.
[14] D. Gusfield, Inference of haplotypes from samples of diploid populations: complexity and algorithms, J. Comput. Biol. 8 (3) (2001) 305–323.
[15] D. Gusfield, Haplotyping as perfect phylogeny: conceptual framework and efficient solutions, in: Proceedings of the Sixth Annual International Conference on Computational Molecular Biology (RECOMB), ACM Press, New York, 2002, pp. 166–175.
[16] E. Halperin, R.M. Karp, Perfect phylogeny and haplotype assignment, in: Proceedings of the Eighth Annual International Conference on Computational Molecular Biology (RECOMB), ACM Press, New York, 2004, pp. 10–19.
[17] G. Kimmel, R. Shamir, The incomplete perfect phylogeny haplotype problem, in: Proceedings of Second RECOMB Satellite Workshop on Computational Methods for SNPs and Haplotypes, Pittsburgh, Pennsylvania, 2004, pp. 59–82.
[18] T. Niu, S. Qin, X. Xu, J. Liu, Bayesian haplotype inference for multiple linked single nucleotide polymorphisms, Amer. J. Human Genet. 70 (1) (2002) 157–169.
[19] N. Patil, A. Berno, D. Hinds, et al., Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21, Science 294 (5547) (2001) 1719–1723.
[20] M. Stephens, N. Smith, P. Donnelly, A new statistical method for haplotype reconstruction from population data, Amer. J. Human Genet. 68 (4) (2001) 978–989.
[21] D.G. Wang, J.B. Fan, C.J. Siao, A. Berno, P.P. Young, et al., Large-scale identification, mapping, and genotyping of single nucleotide polymorphisms in the human genome, Science 280 (5366) (1998) 1077–1082.
[22] J. Zhang, W.L. Rowe, A.G. Clark, K.H. Buetow, Genomewide distribution of high-frequency, completely mismatching SNP haplotype pairs observed to be common across human populations, Amer. J. Human Genet. 73 (5) (2003) 1073–1081.