Proc. IWPEC-04, Vol. 3162 in LNCS, pp. 174-186, Springer, 2004

# Perfect Path Phylogeny Haplotyping with Missing Data is Fixed-Parameter Tractable

Jens Gramm[*], Till Nierhoff[*], and Till Tantau[*]

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

**Abstract.** Haplotyping via perfect phylogeny is a method for retrieving haplotypes from genotypes. Fast algorithms are known for computing perfect phylogenies from complete and error-free input instances—these instances can be organized as a genotype matrix whose rows are the genotypes and whose columns are the single nucleotide polymorphisms under consideration. Unfortunately, in the more realistic setting of missing entries in the genotype matrix, even restricted forms of the perfect phylogeny haplotyping problem become NP-hard. We show that haplotyping via perfect phylogeny with missing data becomes computationally tractable when imposing additional biologically motivated constraints. Firstly, we focus on asking for perfect phylogenies that are paths, which is motivated by the discovery that yin-yang haplotypes span large parts of the human genome. A yin-yang haplotype implies that every corresponding perfect phylogeny *has* to be a path. Secondly, we assume that the number of missing entries in every column of the input genotype matrix is bounded. We show that the perfect path phylogeny haplotyping problem is fixed-parameter tractable when we consider the maximum number of missing entries per column of the genotype matrix as parameter. The restrictions we impose are met by a majority of the problem instances encountered in publicly available human genome data.

## 1 Introduction

### 1.1 Haplotype Inference from Genotypes via Perfect Phylogeny

Single nucleotide polymorphisms (SNPs) are differences in a single base, across the population, within an otherwise conserved genomic sequence. The sequence of SNP states in contiguous SNP positions along a chromosomal region is called a *haplotype*. The knowledge of the haplotypes in the human genome is of particular importance since they are believed to be often linked to medical conditions. However, current technologies suitable for large-scale SNP detection in the human genome—which contains two versions of each chromosome—do not obtain the haplotypes but only the *genotype* information: The genotype specifies, for every SNP position, the two states at this site in the two chromosomes. The genotype contains information only on the combination of SNP states at a given site, but it does not tell us which of the states belongs to which

[*] The authors were supported through DAAD (German academic exchange service) postdoc fellowships.

chromosome. It is an important goal to develop efficient methods for inferring haplotypes from genotypes.

Haplotyping *via perfect phylogeny* is a method for haplotype inference where it is assumed that the (unknown) haplotypes underlying the (observed) genotype data can be arranged in a genetic tree in which each haplotype results from an ancestor haplotype via mutations. The perfect phylogeny approach is popular due to its applicability to real haplotype inference problems and its theoretical elegance. It was introduced by Gusfield [6] and received considerable attention which resulted, among others, in quadratic-time algorithms for the case of complete and error-free input data [1, 2]. In the special case where perfect *path* phylogenies are sought, even a linear time algorithm is known [5].

The main hurdle for current haplotype inference methods is missing input data. Real genotype data usually contain a small fraction of missing data caused by technical problems in the process of genotype detection. In the presence of missing data, haplotyping via perfect phylogeny is NP-hard [8]. This is even true for the restricted case of path phylogenies [5]. In an effort to solve the problem efficiently for restricted cases, Halperin and Karp [7] show that perfect phylogeny haplotyping with missing data is tractable if the input satisfies the ‘rich-data hypothesis’. This hypothesis requires, intuitively, that the data contains enough information to ‘locally’ infer the haplotypes at any two SNP sites. In this paper, we take a different approach which is independent of the rich-data hypothesis. We show that perfect phylogeny haplotyping is fixed-parameter tractable when restricted to path phylogenies and when taking the maximum number of missing entries at a particular SNP site as parameter. Notably, experiments on publicly available genotype data show a significant fraction of the data that allows perfect path phylogenies but fails to satisfy the rich data hypothesis.

## 1.2 Computational Problems

When stripped of the biological context, the haplotype inference problem is a purely combinatorial problem, which we describe in the following. In the combinatorial setting a *haplotype* is a binary string. Each position of the string corresponds to a SNP site. When we observe a certain base at the SNP site, the string contains a 0-entry at the corresponding position; if we observe a certain other base, the string contains a 1-entry. In rare situations one may observe three or even four different bases at a specific SNP site, but these cases are not modeled. A *haplotype matrix* is a binary matrix whose rows are haplotypes.

A *genotype* is a string over the alphabet $\{0,1,2\}$. A 2-entry corresponds to a heterozygous SNP site, meaning that the haplotypes underlying the genotype do not agree at that site. The genotype *resulting* from two haplotypes has a 0-entry or a 1-entry at all positions where both haplotypes agree on a 0-entry or 1-entry, respectively. It has a 2-entry at all positions where they disagree. We say that a genotype matrix $A$ *admits a perfect phylogeny* if there exists a rooted tree $T$, called *perfect phylogeny*, such that:

1. Each column of $A$ labels exactly one edge of $T$.
2. Every edge of $T$ is labeled by at least one column of $A$.
3. For each row $r$ of $A$ there are two nodes in $T$ (possibly identical) labeled $r'$ and $r''$. The labels $r'$ and $r''$ are called *haplotype labels*.

| $A$'s column | 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 |

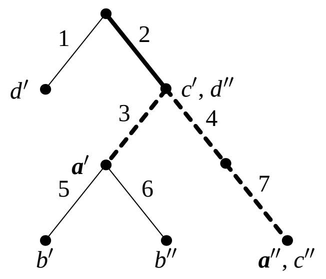

**Fig. 1.** Example of a genotype matrix $A$ and a corresponding perfect phylogeny $T$. The edges of $T$ correspond to the columns of $A$. Each genotype row $r$ in $A$ results from two haplotypes $r'$ and $r''$, which label vertices of the tree. The paths induced by the first row $a$ of $A$ are shown 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 latter path connects the two haplotype vertices $a'$ and $a''$.

4. For every row $r$ of $A$ the set of columns with value 2 in this row forms a path $p$ in $T$ between $r'$ and $r''$. The set of columns with value 1 in this row forms a path from $T$'s root to the top-most node on the path $p$.

An example is depicted in Figure 1. A detailed biological justification for considering perfect phylogenies and for requiring the above properties for the tree $T$ can be found in [5, 6].

Perfect phylogenies with the above properties are called *directed* since 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).

The two versions of the haplotype inference problem via perfect phylogenies mentioned in the previous section are:

*Problem 1.1 (Perfect Phylogeny Haplotyping Problem, {0,1,2}-PPH).*
*Input:* A genotype $\{0,1,2\}$-matrix $A$.
*Question:* Does $A$ admit a perfect phylogeny?

*Problem 1.2 (Perfect Path Phylogeny Haplotyping Problem, {0,1,2}-PPPH).*
*Input:* A genotype $\{0,1,2\}$-matrix $A$.
*Question:* Does $A$ admit a perfect phylogeny that is a path?

It is known that these problems can be solved efficiently, but become computationally intractable if missing data is introduced [5]. Missing entries are denoted by ?-entries in the input matrix. Given such a genotype matrix with ?-entries, we say that it is *pp-realizable* (*ppp-realizable*) if we can replace the question marks with values from $\{0,1,2\}$ such that the resulting genotype matrix admits a perfect (path) phylogeny. This leads to the problem $\{0,1,2,?\}$-PPPH, whose fixed-parameter tractability is the focus of this paper:

*Problem 1.3 (Perfect Path Phylogeny Haplotyping with Missing Entries, {0,1,2,?}-PPPH).*

*Input:* A genotype $\{0,1,2,?\}$-matrix $A$.
*Question:* Is $A$ ppp-realizable?

## 1.3 Biological Relevance of Our Fixed-Parameter Tractability Result

The general perfect phylogeny haplotyping problem with missing entries is NP-complete. In order to solve it, one is thus forced to incorporate additional, biologically motivated constraints on the problem. To this end, firstly, we focus on *path* phylogenies. Secondly, we assume that for each SNP site only a small fraction of genotypes lack information about the state at this site. Thirdly, we focus on *directed* perfect phylogenies.

The first assumption is motivated by the recent discovery [10] that about 75 percent of the human genome can be covered by *yin-yang haplotypes*, i.e., long-region pairs of haplotypes that are heterozygous at every SNP site. In the model of perfect phylogeny haplotyping, the presence of a yin-yang haplotype pair implies that a perfect phylogeny of the haplotypes necessarily has to be a path phylogeny. These findings suggest that an efficient algorithm dealing with path phylogenies can already be applied to a large portion of the genome, leaving the remaining parts for heuristic or more time-intensive postprocessing.

The second assumption is motivated by the expectation that missing data entries should occur only rarely and randomly and should thus not amass at a specific SNP site. Furthermore, a SNP site for which most entries are missing is useless for classification purposes: filling up the numerous missing entries is pure guesswork.

The third assumption is a reasonable one to make since, due to the sparse distribution of missing entries, we can usually identify the labeling of one node that necessarily has to occur in a perfect phylogeny; this can be done using a technique described by Eskin *et al.* [2] (see there for details). Since within each column of the input matrix we are free to exchange 0 and 1 labels, we can choose the labels such that this particular node carries a 0-entry at every position, which makes the perfect phylogeny directed.

We have evaluated all three of our assumptions by considering publicly available genotype data provided by Gabriel *et al.* [3], available from `http://www.broad.mit.edu/mpg/hapmap/hapstruc.html`. The data consist of sets for four populations. Table 1 gives an overview on the distribution of ?-entries in the data. We see that the maximum number of question marks is approximately one fifth of the total column size but also that the average number of question marks per column is much smaller. We can also see from Table 1 that in all except two of the 248 investigated datasets we could identify one of the haplotypes, using the reduction given by Eskin *et al.* [2], thereby making the datasets suitable for our algorithm, which searches for *directed* perfect path phylogenies.

Table 2 treats the assumption of finding perfect phylogenies that are path phylogenies. For a given window length $l$, we computed for every set of $l$ consecutive columns in the input genotype matrix whether these columns give rise to a perfect phylogeny and whether they give rise to a perfect path phylogeny. For our statistic we omitted rows within a window that contain missing entries. It turns out that, on average, approximately two third of the encountered perfect phylogenies are in fact path phylogenies. Notably, less than one third of those genotype matrices admitting a perfect (path) phy-

| Population | A | B | C | D |
|---|---|---|---|---|
| Number of individuals (rows per genotype matrix) | 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 directed genotype matrices | 97% | 100% | 100% | 100% |

**Table 1.** Statistics on the distribution of ?-entries in the human genotype data provided by Gabriel *et al.* [3]. For data from four populations, the table first lists the number of individuals per population whose genotypes were determined. For each individual, the genotypes were determined in 62 different regions of the genome. The table next lists, averaged over the genotype matrices for each population, the maximum and the average number of ?-entries per column. Finally, it lists the percentage of genotype matrices that can be directed.

| Population | Window Size | Genotype Matrices | PP's | PPP's | $\text{RD}_{\text{pp}}$ | $\text{RD}_{\text{ppp}}$ |
|---|---|---|---|---|---|---|
| A | 5 | 3029 | 51% | 40% | 32% | 29% |
| | 8 | 2843 | 29% | 19% | 9% | 7% |
| | 10 | 2721 | 20% | 13% | 3% | 1% |
| B | 5 | 2900 | 35% | 25% | 26% | 21% |
| | 8 | 2720 | 14% | 9% | 7% | 4% |
| | 10 | 2600 | 8% | 5% | 3% | 3% |
| C | 5 | 3001 | 59% | 51% | 36% | 30% |
| | 8 | 2821 | 37% | 28% | 15% | 16% |
| | 10 | 2701 | 27% | 20% | 9% | 9% |
| D | 5 | 2819 | 36% | 26% | 43% | 39% |
| | 8 | 2641 | 14% | 8% | 19% | 16% |
| | 10 | 2525 | 8% | 5% | 7% | 6% |

**Table 2.** Statistics on the frequency of perfect path phylogenies in genotype data provided by Gabriel *et al.* [3]. For each population we slid a window of varying length over the genotype matrices, resulting in the indicated number of genotype matrices that have as many columns as the window size. For these 'narrow' genotype matrices we list the percentage that admits a perfect phylogeny (PP) and a perfect path phylogeny (PPP). We also indicate what percentage of those genotype matrices admitting a perfect phylogeny ($\text{RD}_{\text{pp}}$) or perfect path phylogeny ($\text{RD}_{\text{ppp}}$), respectively, also satisfy the rich-data hypothesis.

logeny satisfy the rich-data hypothesis and, thus, meet the preconditions of the linear-time algorithm given by Halperin and Karp [7].

## 1.4 Our Contribution

In the present paper we show that the haplotype inference problem via perfect phylogeny becomes tractable if we impose the restrictions discussed before: restricting the perfect phylogenies to paths, restricting the number of missing entries per column to a small number, and focusing on directed perfect phylogenies. Due to lack of space, proofs are only given in the full version of this paper.

**Theorem 1.4.** $\{0,1,2,?\}$-PPPH *is fixed-parameter tractable, where the parameter is the maximum number of question marks per column.*

Recall that $\{0,1,2,?\}$-PPPH itself is NP-complete (even if we furthermore require that no 1-entries are present in the genotype matrix) [5]. The loophole, which is exploited in Theorem 1.4, is that the genotype matrices used in the completeness proof are unrealistic insofar as they contain huge amounts of missing data—much more than any real dataset contains.

The algorithm used in the proof of Theorem 1.4 is composed of a powerful preprocessing and a subsequent exponential-time dynamic programming strategy. The preprocessing relies on graph-theoretic arguments and the dynamic programming makes use of concepts from order theory, see for example [9] for an introduction.

Different authors, see for example [6, 4], have shown that there are numerous connections between perfect phylogeny haplotyping problems and more traditional graph realization problems. In [5] it is shown that $\{0,2,?\}$-PPPH is equivalent to a special version of the *interval hypergraph sandwich problem*, which is defined 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 as input given two hypergraphs $H^1 = (V,E^1)$ and $H^2 = (V,E^2)$ 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 1.5 (Interval Hypergraph Sandwich Problem).*
*Input:* Two hypergraphs $H^1 = (V,E^1)$, $H^2 = (V,E^2)$.
*Question:* Is there a hypergraph $H = (V,E)$ sandwiched between $H^1$ and $H^2$ and a linear ordering of $V$ such that each hyperedge $e \in E$ is an interval?

For the special case of the *intersecting interval hypergraph sandwich problem* all hyperedges in $E$ are required to share a common vertex. Even this restricted problem is NP-complete since it is shown in [5] to be equivalent to $\{0,2,?\}$-PPPH. Therefore, our fixed-parameter tractability result for $\{0,1,2,?\}$-PPPH allows us to state the following:

**Theorem 1.6.** *The intersecting interval hypergraph sandwich problem is fixed-parameter tractable, where the parameter is the maximum, taken over all vertices $v$, of edge pairs $(e^1,e^2)$ with $v \in e^2 - e^1$.*

## 2 Fixed-Parameter Algorithm for {0, 1, 2, ?}-PPPH

In this section we present the fixed-parameter algorithm that proves Theorem 1.4. The input of the algorithm is a $\{0,1,2,?\}$-matrix $A$ of dimension $n \times m$ in which there are at most $k$ question marks per column.

### 2.1 Structure of the Algorithm

The algorithm proceeds in two phases. In the first phase, which we call the *preprocessing phase*, the input is first simplified by collapsing multiple columns to one column if we can safely do so. In the second phase dynamic programming is used to compute a completion of the question mark entries such that a perfect path phylogeny results.

The core idea of the preprocessing phase is the following: Suppose several columns become identical by resolving some ?-entries and we replace these columns 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 one. However, the reverse implication is also true if the number of columns that formed the consensus was large enough.

Gusfield [6] introduced the *leaf count*, a 'weight' of a column that depends on the number of 1's and 2's in this column. The main benefit of the preprocessing is that there will be a uniform bound, depending only on $k$, on the number of columns that have the same leaf count. The edges, corresponding to columns, on a path from a leaf to the root in a perfect phylogeny have to carry increasing leaf counts. The core idea of the dynamic programming used in the second phase is to process the columns in order of ascending leaf counts, building a perfect path phylogeny from bottom to top. For each column, we consider all possible ways to resolve the ?-entries in it. For each possible resolution, we check whether a perfect path phylogeny exists that accommodates the resolution and, if so, construct it. The crucial observation, which makes dynamic programming feasible, is the fact that the desired perfect path phylogeny 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.

In the following, we describe the two algorithm phases in more detail. The correctness proofs can be found in the full version of the paper.

### 2.2 Preprocessing Phase

*Terminology.* We start with some technical terminology that will be used in the description of the preprocessing phase.

An *antichain* in a partially ordered set (poset) $P$ is a set of incomparable elements. The *width* of a poset is the size of its largest antichain. An antichain is *maximal* if none of its proper supersets is also an antichain. A maximal antichain is *highest of cardinality $i$* if it has cardinality $i$ and no element of any other maximal antichain of cardinality $i$ strictly dominates any of its elements. We write $\operatorname{hma}_i(P)$ for the set containing the highest maximal antichain of cardinality $i$, if it exists, or for the empty set, if not. Let $\operatorname{hma}(P) := \bigcup_{i=1}^{\infty} \operatorname{hma}_i(P)$. In the following, $\operatorname{hma}_i(P)$ will typically be empty for $i \geq 3$, but it will sometimes be useful to allow for these sets to be nonempty in order to 'catch errors'.

We define a relation $\succ$ on the set $\{0,1,2\}$ by $1 \succ 2 \succ 0$. It is extended to $\{0,1,2\}$-columns by setting $c \succeq c'$ if $c[i] \succeq c'[i]$ for all rows $i$. This extended relation is a poset relation. The *leaf count* $\ell(c)$ of a column $c$ is twice the number of 1-entries plus the number of 2-entries (?- and 0-entries are not counted). For a matrix $A$ let $A_i$ denote the set of columns that have leaf count exactly $i$. Note that if $c \succ c'$ then $\ell(c) > \ell(c')$. Thus the leaf count is a linear extension of the partial order $\succeq$.

A *(partial) resolution* of a column $c$ with ?-entries is a column $c'$ obtained by replacing (some of) the ?-entries with 0, 1, or 2. A *resolution of a set $C$* of columns with ?-entries is a set containing one resolution of each $c \in C$. Let $\operatorname{res} C$ denote the set of all resolutions of a set $C$ of columns and let $\operatorname{pres} C$ denote the set of all partial resolutions.

Since every column has at most $k$ question mark entries, every column has at most $3^k$ resolutions. Thus $|\mathrm{res}\, C| \leq 3^{k|C|}$.

A *consensus* for a set $C$ of columns is a column that is a partial resolution of all $c \in C$. Note that two columns $c$ and $c'$ have a consensus iff $c[i] \neq c'[i]$ 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 $C$.

Our final technical tool is the perfect path phylogeny property (ppp-property), which is defined as follows: Let $C$ be a set of $\{0,1,2\}$-columns We say that $C$ has the *ppp-property* if the following two conditions hold: First, the width of $(C, \succeq)$ is at most two. Second, there are two (possibly empty) chains $(C_1, \succeq)$ and $(C_2, \succeq)$ covering $(C, \succeq)$ such that for each row the following holds: If some column in one of the two chains has a 1-entry in that row, all columns in the other chain must have a 0-entry in that row. The name of the ppp-property is justified by the following lemma, see [5].

**Lemma 2.1.** *A $\{0,1,2\}$-matrix A admits a perfect phylogeny that is a path iff the set of columns of A has the ppp-property.*

*Algorithm.* The objective of the preprocessing is to ensure that, during the actual algorithm, there will only be a fixed number of columns that have any given leaf count. Normally, if a genotype matrix without missing data and without duplicate columns has more than two columns of the same leaf count, the matrix does not permit a perfect phylogeny that is a path. The reason is that different columns with the same leaf count are always incomparable with respect to the partial order $\succ$ and thus cannot lie on the same path from the root to a leaf of a perfect phylogeny—and if the perfect phylogeny is a path, there are only two paths starting at the root.

In the presence of missing data there can be an arbitrary number of different columns that have the same leaf count, but still the genotype matrix allows a perfect path phylogeny. The reason is that we might be able to 'collapse' all the different columns into one column by filling up the missing entries appropriately.

The idea of the preprocessing step is to perform such collapsing, but we cannot just 'collapse everything that can be collapsed' as it may be necessary *not* to collapse columns in order to realize the matrix. We show that it is 'safe' to collapse columns in the input matrix if their number is large. From here on, we denote $\kappa(i) := 6^i \cdot i!$.

**Lemma 2.2.** *Let A be a genotype matrix with at most k question mark entries per column. Let C be a set of columns in A and let $\ell \geq 1$ be minimal such that the columns in C have a consensus c containing $k - \ell$ questions mark entries. If $|C| > \kappa(\ell)$, then the following holds: The matrix B obtained from A by replacing all columns in C by c is ppp-realizable iff B is realizable.*

To give a flavor of the proof, which is omitted due to lack of space, consider the case $\ell = 1$ and consider seven columns in $A$ that have a consensus $c$. We are done if there is a ppp-realization of $A$ that contains the consensus $c$. So suppose that this is not the case and that the realizations of the seven columns of $A$ *differ* from the consensus.

Since $\ell = 1$, for each of the seven columns there is exactly one row where the column differs from the consensus, namely by having a ?-entry in that *disputed* position.

We extract the rows containing disputed positions. Since any two of the seven columns are different, every disputed position must be in a different row. Thus we extract at least seven rows. For any seven rows, at least three rows must have a 0-entry, a 1-entry, or a 2-entry in the consensus. We extract these three rows and sort them according to the row that contains the ?-entry, resulting in a matrix that looks this: $\begin{pmatrix} ? & x & x \\ x & ? & x \\ x & x & ? \end{pmatrix}$ with $x \in \{0,1,2\}$. This matrix, and hence the original one, is not ppp-realizable, unless one of the ?-entries is resolved by $x$.

The rest of the lemma is proved is proved by induction on $\ell$. For the inductive step, we use graph-theoretic arguments on a graph defined on the columns of the genotype matrix $A$ and combine them with a contradictory argument similar to the one just given.

While Lemma 2.2 allows us to perform collapses whenever numerous columns have a common consensus, it is not clear how we can *find* these columns efficiently. Given a genotype matrix $A$ in which every column has at most $k$ question marks, we call a set $C$ of columns from $A$ a *collapsible* set if it meets the requirements of Lemma 2.2, i.e., if there exists a consensus $c$ with exactly $\ell$ question mark entries and if $|C| > \kappa(\ell)$. The following lemma states that a *collapsible* set of columns can be found efficiently.

**Lemma 2.3.** *Given an $n \times m$ genotype matrix $A$ with at most $k$ question mark entries per column, we can, in time $O(4^k m^2 n)$ and space $O(mn)$, find a collapsible set $C$ of columns with its corresponding consensus $c$ or determine that no such set exists.*

In the preprocessing step we remove duplicate columns and then invoke the algorithm from Lemma 2.3 as often as possible. As long as the algorithm outputs some set $C$ and a consensus $c$, we collapse the columns in $C$, remove duplicates, and repeat. The resulting preprocessing is summarized in Figure 2.

The essential property of the preprocessed matrix is that Lemma 2.2 'does not apply' to it. Thus the dimension of any column $c$ in it is at most $\kappa(k) = 6^k \cdot k!$. This allows us to sum up the effect of the preprocessing phase in the following theorem, which will be crucial for keeping the dynamic programming tables of our main algorithm 'small'. From here on, we use $\lambda(i) := (4i+2)\kappa(i)$.

**Theorem 2.4.** *Let $B$ result from $A$ by the preprocessing algorithm in Figure 2. Then $B$ is ppp-realizable iff $A$ is. Furthermore, if more than $\lambda(k)$ columns in $B$ have the same leaf count, then both $A$ and $B$ are not ppp-realizable.*

*Running time.* By Lemma 2.3, finding a set of columns that can be collapsed (realized inside the repeat-loop in Figure 2) can be done in time $O(4^k m^2 n)$. Since we can perform at most $m$ collapsing steps before running out of columns, the preprocessing algorithm will stop after at most $O(4^k m^3 n)$ steps.

## 2.3 Dynamic Programming Phase

*Terminology.* In the following, we introduce terminology for the description of the dynamic programming table. The input for the dynamic programming phase is a pre-

```
Input:  Genotype matrix $A$ with at most $k$ question mark entries per column.
Output: Genotype matrix $B$ in which every column has dimension at most $\kappa(k)$ and
        which is ppp-realizable iff $A$ is ppp-realizable.

$B \leftarrow A$
repeat
  for $\ell \leftarrow 1$ to $k$ do
    foreach column $b$ of $B$ do
      foreach $p \in \mathrm{pres}\{b\}$ do
        if $p$ has $k - \ell$ many ?-entries then
          $C \leftarrow \{c \mid c$ is a column of $B$ and a resolution of $p\}$
          if $|C| > \kappa(l)$ then
            $B \leftarrow B$ with all columns in $C$ replaced by $p$
            break for loop
until $B$ remains unchanged
return $B$
```

**Fig. 2.** The preprocessing algorithm

processed $n \times m$ matrix $B$. Observe that the leaf counts of the columns of $B$ range between 0 and some number $L \leq 2n$. During the execution of our algorithm, for increasing numbers $i \in \{0, \ldots, L\}$ we consider the set $B_i$ of columns of $B$ that have leaf count $i$. By Theorem 2.4, if the size of $B_i$ is larger than $\lambda(k)$, we can output that no realizable resolution exists.

In the following, we define sets $\mathcal{R}_i$ that, intuitively, contain all ppp-realizable candidate resolutions for the columns whose leaf count 'could become' $i$. Consider the union $B_{[i-2k,i]}$ of the sets $B_{i-2k}$, $B_{i-2k+1}$, ..., $B_i$ (we assume that $B_j := \emptyset$ for negative $j$). Thus, $B_{[i-2k,i]}$ contains all columns whose leaf count, after resolution of the question marks, could become $i$. With Theorem 2.4, the cardinality of $B_{[i-2k,i]}$ is at most $(2k+1)\lambda(k)$. We require that a set $R \in \mathrm{res}\, B_{[i-2k,i]}$ is in $\mathcal{R}_i$ if $(R, \succeq)$ has the ppp-property:

$$\mathcal{R}_i := \{R \mid R \in \mathrm{res}\, B_{[i-2k,i]},\, R \text{ is ppp}\}.$$

We can bound the size of $\mathcal{R}_i$ by bounding the size of $\mathrm{res}\, B_{[i-2k,i]}$: each question mark in a column in $B_{[i-2k,i]}$ can be resolved in three different ways and there are at most $k(2k+1) \cdot \lambda(k)$ question marks altogether in $B_{[i-2k,i]}$. Thus, $|\mathcal{R}_i| \leq 3^{k(2k+1)\lambda(k)}$.

We next extend our notion of candidate resolutions which are so far defined for a set of columns having a limited range of leaf counts. We introduce sequences of these candidate resolutions covering all columns having leaf counts up to $i$. Let us call a sequence $(R_0, \ldots, R_i) \in \mathcal{R}_0 \times \cdots \times \mathcal{R}_i$ *consistent* if every column of $B$ is resolved in exactly the same way in all $R_j$ in which it occurs. Let

$$\mathcal{R}_{\leq i} := \{(R_0, \ldots, R_i) \in \mathcal{R}_0 \times \cdots \times \mathcal{R}_i \mid (R_0, \ldots, R_i) \text{ consistent}, R_0 \cup \cdots \cup R_i \text{ is ppp}\}.$$

Intuitively, each $(R_0, \ldots, R_i) \in \mathcal{R}_{\leq i}$ (which does not contain ?-entries) is a possible 'consistent' realization of columns in $B_0, \ldots, B_i$ (which may contain ?-entries). The

**Input:** Genotype matrix $A$ with at most $k$ question mark entries per column.
**Output:** Statement whether $A$ admits a perfect path phylogeny.

```
$B \leftarrow \mathit{preprocess}\, A$
**for** $i \leftarrow 0$ **to** $L$ **do**
    **if** $|B_i| > \lambda(k)$ **then output** '$A$ has no realizable resolution'**; stop**
$j \leftarrow \min\{j \mid B_j \neq \emptyset\}$
**foreach** $R_j \in \mathcal{R}_j$ **do**
    **if** $R_j$ is ppp **then** $\mathcal{H}(R_j) \leftarrow \{\mathrm{hma}(R_j)\}$
**for** $i \leftarrow j+1$ **to** $L$ **do**
    **if** $B_{[i-2k,i]} \neq \emptyset$ **then**
        $j \leftarrow \max\{j \mid j < i$ and $B_{[j-2k,j]}$ is not empty$\}$
        **foreach** $R_i \in \mathcal{R}_i$ **do**
            $\mathcal{H}(R_i) = \{\mathrm{hma}(H_j \cup [R_i]^{=i}) \mid R_j \in \mathcal{R}_j, (R_j, R_i)$ is consistent,
                                   $H_j \in \mathcal{H}(R_j), H_j \cup [R_i]^{\geq i}$ is ppp$\}$
**if** $\mathcal{H}(R_L) \neq \emptyset$ **then output** '$A$ admits a perfect path phylogeny'
                 **else output** '$A$ does not admit a perfect path phylogeny'
```

**Fig. 3.** Main algorithm. Using standard dynamic programming techniques, the above algorithm can be modified to output a realizable resolution of $A$ instead of just deciding whether such a resolution exists.

columns in $R_0 \cup \cdots \cup R_i$, however, may have leaf counts larger than $i$, namely up to $i+2k$. Being interested in 'filtering out' those columns that actually have a certain leaf count, for a set $R$ of columns we define the set $[R]^{=j}$ to contain the those columns in $R$ that have leaf count $j$. In the same way we define $[R]^{\leq j}$ and $[R]^{>j}$. Thus, we use subscripts to refer to the leaf count *before* replacing question marks, and superscripts to refer to the leaf count *after* the resolution.

Our final technical definition concerns the highest maximal antichains of the column sets. For $R_i \in \mathcal{R}_i$ we define:

$$\mathcal{H}(R_i) := \left\{\mathrm{hma}\left([R_0 \cup \cdots \cup R_i]^{\leq i}\right) \mid (R_0, \ldots, R_i) \in \mathcal{R}_{\leq i}\right\}.$$

*Algorithm.* The algorithm uses dynamic programming. The employed dynamic programming table has one column for every leaf count ranging from 0 to $L$. The column corresponding to leaf count $i$ has one entry for every $R_i \in \mathcal{R}_i$. The table entry corresponding to $R_i \in \mathcal{R}_i$ is defined to contain $\mathcal{H}(R_i)$. The dynamic programming step computes $\mathcal{H}(R_i)$ based only on $\mathcal{H}_j(R_j)$ for all $R_j \in \mathcal{R}_j$, where $j < i$ is maximal such that $B_{[j-2k,j]}$ is not empty. Figure 3 shows pseudo-code for solving $\{0,1,2,?\}$-PPPH. Proving the correctness of this algorithm is deferred to the full version of the paper.

*Running time.* The algorithm iterates $L \leq 2n$ times. In each iteration it computes $\mathcal{H}(R_i)$ for each $R_i \in \mathcal{R}_i$. For computing $\mathcal{H}(R_i)$, we consider each $\mathcal{R}_j$ and, for each of these, each $H_j \in \mathcal{H}(R_j)$. We know that both $|\mathcal{R}_j|$ and $|\mathcal{R}_i|$ are bounded by $3^{k(2k+1)\lambda(k)}$ (see definition of $\mathcal{R}_i$). The size of $H_j$ can be bounded by $3^{k(2k+1)\lambda(k)} \cdot 2n$. The test whether

$(R_j, R_i)$ is consistent and the test whether $H_j \cup R_i$ has the ppp-property can both be done in time $O(|R_i|)$. In summary, the running time is $O\big(4^k m^3 n + 3^{O(k^3 \cdot 6^k \cdot k!)} \cdot n^2\big)$, including the preprocessing.

## 3 Conclusion and Open Problems

We have shown that the haplotype inference problem via perfect phylogeny in the presence of missing data becomes feasible if the input data meets two conditions: the number of missing entries per SNP site is small and the perfect phylogeny is a path. An analysis of publicly available data shows that these requirements are often met by genomic data available for humans.

While, admittedly, the factor in the running time of our algorithm that depends on $k$ grows quickly with $k$, the estimate that we give is a worst-case estimate that might be far too pessimistic in real instances.

Two main open questions remain concerning the presented problem parameterization, namely by the maximum number of missing SNP states per SNP site. First, is the undirected version of the perfect path phylogeny problem fixed-parameter tractable? Second, and more importantly, is the perfect (possibly non-path) phylogeny case fixed-parameter tractable?

## References

1. V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph. Haplotyping as perfect phylogeny: A direct approach. *Journal of Computational Biology*, 10(3–4):323–340, 2003.
2. 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.
3. 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, and D. Altshuler. Structure of halpotype blocks in the human genome. *Science*, 296:2225–2229, 2002.
4. M. C. Golumbic and A. Wassermann. Complexity and algorithms for graph and hypergraph sandwich problems. *Graphs and Combinatorics*, 14:223–9, 1998.
5. J. Gramm, T. Nierhoff, R. Sharan, and T. Tantau. On the complexity of haplotyping via perfect phylogeny. In *Proceedings of the 2nd RECOMB Satellite Workshop on Computational Methods for SNPs and Haplotypes*, LNBI. Springer, 2004. To appear.
6. D. Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. In *Proceedings of the 6th RECOMB*, pages 166–75. ACM Press, 2002.
7. E. Halperin and R. M. Karp. Perfect phylogeny and haplotype assignment. In *Proceedings of the 8th RECOMB*, pages 10–19. ACM Press, 2004.
8. Gad Kimmel and Ron Shamir. The incomplete perfect phylogeny haplotype problem. In *Proceedings of the 2nd RECOMB Satellite Workshop on Computational Methods for SNPs and Haplotypes*, LNBI. Springer, 2004. To appear.
9. W. T. Trotter. *Combinatorics and Partially Ordered Sets: Dimension Theory*. The Johns Hopkins University Press, Baltimore, 1992.
10. 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.