Influence of Tree Topology Restrictions on the Complexity of Haplotyping with Missing Data

Michael Elberfeld Ilka Schnoor Till Tantau

Institut für Theoretische Informatik
Universität zu Lübeck
D-23538 Lübeck, Germany
{elberfeld,schnoor,tantau}@tcs.uni-luebeck.de

Technical Report
SIIM-TR-A-08-05
Schriftenreihe der Institute für Informatik/Mathematik der
Universität zu Lübeck

September 12, 2008

Abstract

Haplotyping, also known as haplotype phase prediction, is the problem of predicting likely haplotypes based on genotype data. One fast haplotyping method is based on an evolutionary model where a *perfect phylogenetic tree* is sought that explains the observed data. Unfortunately, when data entries are missing, as is often the case in real laboratory data, the resulting formal problem IPPH, which stands for incomplete perfect phylogeny haplotyping, is NP-complete and no theoretical results are known concerning its approximability, fixed-parameter tractability or exact algorithms for it. Even radically simplified versions, such as the restriction to phylogenetic trees consisting of just two directed paths from a given root, are still NP-complete, but here, at least, a fixed-parameter algorithm is known. We generalize this algorithm to arbitrary tree topologies and present the first theoretical analysis of an algorithm that works on arbitrary instances of the original IPPH problem. At the same time we also show that restricting the tree topology does not always make finding phylogenies easier: while the incomplete directed perfect phylogeny problem is well-known to be solvable in polynomial time, we show that the same problem restricted to path topologies is NP-complete.

1 Introduction

Biological Background and Previous Results.

Haplotype phase prediction is an important preprocessing step in genomic disease and medical condition association studies. In these studies two groups of people are identified, where one group has a certain disease or medical condition while the other has not, and one tries to find correlations between group membership and the genomic data of the individuals in the groups. The genomic data typically consists of information about which bases are present in an individual's DNA at so-called SNP sites (single nucleotide polymorphism sites). While the DNA sequences of different individuals are mostly identical, at SNP sites there may be variations. Low-priced methods for large-scale inference of genomic data can read out, separately for each SNP site, the bases present, of which there can be two since we inherit one chromosome from our father and one from our mother. However, since the bases at different sites are determined independently, we have no information to which chromosome a base belongs to. For *homozygous sites,* where the same base is present on both chromosomes, this is not a problem, but for *heterozygous sites* this information, called the *phase* of an SNP site, is needed for accurate correlations.

The idea behind *haplotype phase prediction* or just *haplotyping* is to computationally predict likely phases based on the laboratory data (which misses this information). For an individual, the genomic input data without phase information is called the *genotype* while the two predicted chromosomes are called *haplotypes*.

From a mathematical point of view, haplotypes can be conveniently coded as strings over the alphabet $\{0,1\}$, where for a given site 0 stands for one of the bases that can be observed in practice, while 1 encodes a second base that can also be observed. The case that three bases are observed happens so seldom that it can be ignored. A genotype g is, conceptually, a sequence of sets that arises from two haplotypes h_1 and h_2 as follows: The ith set in the sequence g is $\{h_1[i], h_2[i]\}$. However, it is customary to encode the set $\{0\}$ as 0, to encode $\{1\}$ as 1, and $\{0,1\}$ as 2, so that a genotype is actually a string over the alphabet $\{0,1,2\}$. For example, the two haplotypes 0110 and 0101 give rise to (we also say *explain*) the genotype 0122; and so do 0100 and 0111.

Since different haplotype pairs can explain the same genotype and any single haplotype is equally likely a priori, haplotyping is not possible if only a single genotype is given. However, if a whole set of genotypes from a larger group of different individuals is given, certain sets of haplotypes that explain these genotypes are more likely than others. For instance, a small set of explaining haplotypes is more likely than a large set since haplotypes mutate only rarely. It is customary to formalize sets of genotypes as matrices (each row is a genotype) and also sets of explaining haplotypes (each row contains a haplotype and rows $2i-1$ and $2i$ of the haplotype matrix explain exactly the genotype in row i of the genotype matrix).

One important method of haplotyping is based on the *perfect phylogeny approach* proposed by Gusfield [16]. The idea is to seek a haplotype matrix that explains the genotype matrix and whose rows (which are the haplotypes) can be arranged in a *perfect phylogenetic tree*. This means the following: A haplotype matrix B *admits a perfect phylogeny* if there exists a tree (an undirected, connected, acyclic graph) T_B such that:

1. Each row of B labels exactly one node of T_B.
2. Each column of B labels exactly one edge of T_B.
3. Each 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 $h_1[i] \neq h_2[i]$ if, and only if, i lies on the path from h_1 to h_2 in T_B.

When a haplotype matrix B admits a perfect phylogeny T_B and, at the same time, explains a genotype matrix A, we also say that A *admits a perfect phylogeny* (namely T_B). The formal *perfect phylogeny haplotyping problem* (PPH) is then the set of all genotype matrices that admit a perfect phylogeny. Gusfield [16] showed that PPH is solvable in polynomial time. However, in practice, laboratory data is never perfect and some entries may be missing in the input genotype matrices. In this case, the input matrices may contain ?-entries in addition to the 0-, 1-, and 2-entries. The objective is then to replace the missing entries by normal entries such that the resulting matrix is in PPH. This problem is known as IPPH, where the I stands for *incomplete* (in the following, when we prefix a problem with the letter I, we mean that the input matrix may contain ?-entries and the objective is to fill them up so that the resulting matrix is an instance of the problem without the I). Unfortunately, IPPH is NP-complete [25]. A heuristic is known for solving it [24], but no guarantees can be made concerning its runtime.

In order to solve the problem, one can try to exploit properties of typical input data that may make the problem easier. The first simplification is the notion of *directedness*. In real data, typically some genotype is completely known and is completely homozygous, which means that one haplotype of the sought haplotype matrix is already known. Since the roles of 0-entries and 1-entries can be exchanged individually for each column, we may assume that the known haplotype is the all-0-haplotype. This problem variant is called "directed" because the position of the all-0-haplotype in the phylogenetic tree singles out a node, which is then considered the root of the tree and gives an orientation to the

tree. The resulting problem is called IDPPH, with D generally standing for "directed," but it is still NP-complete [20].

A second, rather radical simplification (which is nevertheless often backed by the data) was proposed by Gramm et al. [13]: In addition to being directed, they require the (undirected) phylogenetic tree to form a simple path and call the problem incomplete directed perfect *path* phylogeny haplotyping. This problem is *still* NP-complete, but Gramm et al. show that there is a fixed-parameter algorithm, where the parameter is the maximum number of ?-entries per column.

Another radical simplification is to study matrices in which no heterozygous sites are found. This is same as getting already phased haplotypes as input and the question is just whether they are arrangeable in a perfect phylogeny. So, for the problem IPP (note the missing H, since no haplotyping needs to be done) we get an incomplete haplotype matrix and must decide whether the missing entries can be filled up with 0-entries and 1-entries so that the completed matrix admits a perfect phylogeny. This problem is still NP-complete. However, the directed variant IDPP is solvable in polynomial time [2, 22].

In the present paper we further the study of the computational complexity of IPPH and the above variants. We are especially interested in the following question: How do restrictions on the tree topology influence the complexity of the problem? In other words, what is the complexity of $\text{IPPH}_{\text{leafs}\leq l}$, where the explaining phylogenetic tree may have at most l leafs. As stated above, the best known result is that $\text{IDPPH}_{\text{leafs}\leq 2}$ is still NP-complete, but lies in FPT.

Our Results. Our first main result, presented in Section 2, is a completeness result.

Theorem 1.1. $\text{IDPP}_{\text{leafs}\leq l}$ *is* NP*-complete for every* $l \geq 2$.

We found this result astonishing because IDPP $\in$ P. In other words, in contrast to all experience from previous results, see the section on related work below, restricting the tree topology makes IDPP harder, not easier. Naturally, there are other examples: Finding a spanning tree for a graph is easy, finding a spanning path is hard.

The problem $\text{IDPP}_{\text{leafs}\leq l}$ reduces to many other problems. It is easy to see (but not trivial) that a problem where the input matrix consists of (possibly incomplete) haplotypes reduces to the same problem for (possibly incomplete) genotype matrices via the identity mapping. Thus, the first result implies that $\text{IDPPH}_{\text{leafs}\leq l}$ is also NP-complete, which was previously proved by Gramm et al. [13] for $l = 2$. It is also easy to see that any directed problem reduces to the undirected version by adding an all-0-row. Thus, $\text{IPP}_{\text{leafs}\leq l}$ is also NP-complete. Indeed, all previously known NP-completeness results for variants of IPPH follow from the above theorem, except for the NP-completeness of IPP.

The completeness result also sparked our interest in trying to find an even more restricted version of IPPH that is still NP-complete. A candidate would be $\text{IDPPH}_{\text{sorted}}$, where the "phylogenetic tree" must be a single directed path. We show, however, that this problem is NL-complete. For IDPP we show that it is also NL-hard, but were not able to prove any better upper bound than the known bound IDPP $\in$ P.

Our second main result, presented in Section 3, is an algorithm for solving IPPH that allows a rigorous runtime analysis.

Theorem 1.2. *Given an incomplete genotype matrix A, we can decide whether* $A \in$ IPPH *holds in time* $f(k,l)n^2m^{O(l)}$. *Here k is the maximal number of ?-entries per column, n is the number of rows in the incomplete genotype matrix, m is the number of columns, and f is some function. The parameter l is the minimal number of leafs any perfect phylogeny admitted by a completion of A can have (set* $l = m$ *if no such perfect phylogeny exists).*

This is, in essence, a statement about the fixed-parameter tractability of IPPH. It says that IPPH lies in the class XP for the parameter pair (k,l). More importantly, it says that for each fixed l, the problem $\text{IPPH}_{\text{leafs}\leq l}$ is fixed-parameter tractable with respect to the number of unknown entries per column. This settles the central open problem of [13], namely whether the result that $\text{IDPPH}_{\text{leafs}\leq 2}$ is fixed-parameter

tractable also holds for the undirected case and for larger number of leafs. On both accounts, we answer this question affirmatively.

Related Work. Haplotyping methods can be split into two groups: Statistical [8, 9, 18] and combinatorial. There are two main combinatorial methods: Maximum parsimony haplotyping [3, 4, 10, 15, 19] and the more recent perfect phylogeny approach that was introduced by Gusfield [16] and later explored by numerous authors [1, 7, 5, 21, 13, 23, 6].

The idea of considering restricted tree topologies to speed up haplotyping is due to Gramm et al. [13] and was recently also investigated in the context of finding block partitions [11]. A different approach to deal with the NP-completeness of IPPH is due to Halperin and Karp [17]. They present a polynomial-time algorithm for IPPH that works for special instances satisfying the so-called "rich data hypothesis."

For complete data, numerous results on the complexity of PPH and its variants are known. Gusfield showed that the problem can be solved in polynomial time [16], further papers first presented simpler polynomial-time algorithms [1, 7] and later even linear-time algorithms [5, 21, 23]. In [6] we have shown that PPH is hard for logarithmic space and lies in NC^2.

The influence of restricting the tree topology on the complexity of haplotyping problems has, prior to the present paper, always been begin: In [13] it is shown that $\text{IDPPH}_{\text{leafs}\leq 2}$ has a fixed-parameter algorithm, which is not known to be the case for IPPH. In [13] it is shown that partitioning a complete genotype matrix into a minimal number of column sets such that each set admits a perfect *path* phylogeny is equivalent, in complexity theoretic terms, to finding maximal matchings; while the same problem for arbitrary perfect phylogenies is NP-hard and even very hard to approximate. Finally, in [6] it is shown that $\text{DPPH}_{\text{leafs}\leq 2}$ lies in AC^0, while DPPH is L-hard.

2 Hardness Results

In this section we present our completeness and hardness results for perfect phylogeny and haplotyping problems. Before we detail these results, let us first review the notation that we are going to use: For the basic problem PP the input is a (complete) haplotype matrix and the question is whether it admits a perfect phylogeny. This basic problem can be modified by adding prefixes and indices: We add the prefix D to indicate that one of the nodes in the phylogeny must be labeled with the all-0-haplotype. We append H to indicate that the input is a genotype matrix rather than a haplotype matrix and an explaining haplotype matrix admitting a perfect phylogeny must be found. We add the prefix I to indicate that ?-entries may be present in the input. We add the index "leafs $\leq l$" to indicate that only perfect phylogenies having at most l leafs are allowed (a leaf is a degree 1 node in the undirected graph underlying the perfect phylogeny). Finally, we use the index "sorted" to indicate that the phylogeny must form a path and that the all-0-haplotype (the root) must label one of its ends.

We first show that the phylogenetic sorting problems $\text{IDPP}_{\text{sorted}}$ and $\text{IDPPH}_{\text{sorted}}$ are complete for NL. In particular, these problems are not NP-complete under standard assumptions. These results can be used to show that IDPP is hard for NL. Previously, it was already known that we need at least linear-time to solve IDPP [14].

We then prove our first main result, Theorem 1.1 from the introduction, which states that $\text{IDPP}_{\text{leafs}\leq l}$ is NP-complete. Note that for $l = 2$ the only difference between the NL-complete $\text{IDPP}_{\text{sorted}}$ and the NP-complete $\text{IDPP}_{\text{leafs}\leq 2}$ is that in the latter problem the root can be anywhere on the phylogenetic path, whereas for the first problem it must lie at one of the ends. Also note that IDPP is known to lie in P.

All reductions in this paper are first-order many-one-reductions.

2.1 Hardness and Completeness For NL

Theorem 2.1. $\text{IDPP}_{\text{sorted}}$ *and* $\text{IDPPH}_{\text{sorted}}$ *are complete for* NL.

Proof. We first review the partial order $\succ$ on the columns of genotype matrices that was introduced by Eskin, Halperin and Karp [7] and later used in [13]. It will allow an easy characterization of haplotype and genotype matrices that admit phylogenetic sortings. Let $1 \succ 2 \succ 0$. For two columns c_1 and c_2 of a complete genotype matrix, let $c_1 \succeq c_2$ if $c_1[i] \succeq c_2[i]$ holds for each row i. We say that two columns c_1 and c_2 are *comparable* if $c_1 \succeq c_2$ or $c_2 \succeq c_1$; otherwise, they are *incomparable*. A column c_1 *dominates* a column c_2 if $c_1 \succeq c_2$ and $c_1 \neq c_2$. Using this partial order, we can characterize phylogenetic sortings as follows:

Claim. *A genotype matrix A admits a phylogenetic sorting if, and only if, every pair of columns of A is comparable.*

Proof. For the "if"-direction let A be an $n \times m$ genotype matrix with pairwise comparable columns. Then there exists a sorting $c_1 \succeq \cdots \succeq c_m$ of the columns of A. We construct a haplotype matrix B for A that admits a phylogenetic sorting as follows: Let g be a genotype from A. Since the order $\succeq$ is defined componentwise, there exist indices i and j such that g contains its 1-entries in columns $c_1, \ldots, c_i$, its 2-entries in columns $c_{i+1}, \ldots, c_j$, and its 0-entries in the remaining columns $c_{j+1}, \ldots, c_n$. We set the explaining haplotypes for g to $1^i 1^{j-i} 0^{n-j}$ and $1^i 0^{n-i}$. In the phylogenetic sorting for B, the sequence of edge labels is $c_1, \ldots, c_m$. For the "only if"-part consider a genotype matrix A with two incomparable columns c_1 and c_2. Each explaining haplotype matrix for A contains the submatrix $\left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ in c_1 and c_2. No phylogenetic sorting exists for this submatrix and, hence, the whole matrix does not admit a phylogenetic sorting. □

We next introduce a directed graph that can be thought of as a generalization of the order $\succeq$ to incomplete genotype matrices. For an $n \times m$ genotype matrix A, the directed graph G_A has the columns $c_1, \ldots, c_m$ of A as its vertices. There is an edge from a column c_i to a column c_j if there is a row k with $c_i[k] \succ c_j[k]$. The idea behind this definition is that c_i dominates c_j in each completion of A with pairwise comparable columns. This leads to the following characterization of $\text{IDPPH}_{\text{sorted}}$:

Claim. *An incomplete genotype matrix A lies in* $\text{IDPPH}_{\text{sorted}}$ *if, and only if, G_A is acyclic.*

Proof. For the "if"-direction let A be an $n \times m$ incomplete genotype matrix such that G_A is acyclic. The graph G_A induces a topological sorting $c_1, \ldots, c_m$ on the columns. We derive completions $c'_1, \ldots, c'_m$ for them in a stepwise fashion such that after step i the columns $c'_1, \ldots, c'_i$ are pairwise comparable. In step i, consider the column c_i. For each row k with $c_i[k] \neq ?$, set $c'_i[k] = c_i[k]$. For each row k with $c_i[k] = ?$, set $c'_i[k] = 1$ if there is a $j > i$ with $c_j[k] = 1$ and, otherwise, set $c'_i[k] = 2$. The completion assures that after an iteration i there is no $j > i$ and row k, with $c_j[k] \succ c'_i[k]$. We claim that after each iteration i the columns $c'_1, \ldots, c'_i$ are sorted in nonincreasing order. This trivially holds after the first iteration. If it holds after iteration i, it also holds after iteration $i+1$ where column c_{i+1} is completed: If an entry $c_{i+1}[k]$ is already known, the relation $c_{i+1}[k] \succ c'_i[k]$ is not possible, as argued above. If an entry $c_{i+1}[k]$ is not known, the procedure chooses a value that does not exceed $c_i[k]$. Overall, we obtain that there is a completion of A with pairwise comparable columns and we know $A \in \text{IDPPH}_{\text{sorted}}$ by the first claim.

For the "only if"-direction, let $c_1, \ldots, c_l$ be a cycle in G_A. We assume, for the sake of contradiction, that there are pairwise comparable completions $c'_1, \ldots, c'_l$. Since for each i there exists a row k with $c_i[k] \succ c_{i+1}[k]$ and also $c_l[k] \succ c_1[k]$ for some row k, we have $c'_1 \succ \cdots \succ c'_k \succ c'_1$, which contradicts the partial order. □

To prove the theorem, we now show that the problems ACYCLIC, $\text{IDPP}_{\text{sorted}}$, and $\text{IDPPH}_{\text{sorted}}$ are equivalent. For the NL-complete problem ACYCLIC we are given a directed graph without loops and ask

whether the graph is acyclic. First, we reduce ACYCLIC to $\text{IDPP}_{\text{sorted}}$: For a directed graph $G = (V,E)$ we construct an $|E| \times |V|$ incomplete haplotype matrix B: We identify the vertices with columns and the edges with rows. For each directed edge $(v,w) \in E$, we set the corresponding row in the column of v to 1, in the column of w to 0 and the remaining entries to ?-entries. Then G is isomorphic to G_B and since haplotype matrices are special cases of genotype matrices, by the above claim the construction is a reduction. Second, $\text{IDPP}_{\text{sorted}}$ reduces to $\text{IDPPH}_{\text{sorted}}$ via the identify mapping. Third, for the reduction from $\text{IDPPH}_{\text{sorted}}$ to ACYCLIC, we construct the graph G_A for a given incomplete genotype matrix A. Again by the above claim, this construction is a reduction. □

Theorem 2.2. IDPP *is* NL*-hard.*

Proof. We present a reduction from $\text{IDPP}_{\text{sorted}}$ to IDPP. Since $\text{IDPP}_{\text{sorted}}$ is NL-hard by Theorem 2.1, the claim follows. Given an incomplete haplotype matrix B, our reduction constructs the matrix B' by appending the row $1 \dots 1$. We claim that $B \in \text{IDPP}_{\text{sorted}}$ if, and only if, $B' \in \text{IDPP}$. For the "only-if"-part let there exist a completion of B that admits a phylogenetic sorting. The additional all-1-row clearly does not change this property and, thus, B' admits a perfect phylogeny. For the "if"-direction let $T_{B'}$ be a directed perfect phylogeny for a completion of B'. We argue that $T_{B'}$ is already a phylogenetic sorting: The root node of $T_{B'}$ is labeled by the all-0-row and one node is labeled by the all-1-row. All columns lie on the path between these nodes. By contracting the edges of $T_{B'}$ that are not labeled by columns, we obtain a phylogenetic sorting for a completion of B'. Since B is a submatrix of B', we also know $B \in \text{IDPP}_{\text{sorted}}$. □

2.2 Completeness for NP

In the present section we prove our first main result, Theorem 1.1 from the introduction: $\text{IDPP}_{\text{leafs}\leq l}$ *is* NP*-complete for each* $l \geq 2$. Since IDPP $\in$ P, this is the first time a perfect *path* phylogeny problem is harder than the corresponding problem for general perfect phylogenies. The ideas of our reduction are extensions of the NP-hardness proof for $\text{IDPPH}_{\text{leafs}\leq 2}$ from Gramm et al. [13].

Proof of Theorem 1.1. We only show hardness. Fix an $l \geq 2$. We present a reduction from the problem MONOTONE NAE3SAT to $\text{IDPP}_{\text{leafs}\leq l}$. For this problem we are given a propositional formula in conjunctive normal form with three positive literals per clause (monotone 3CNF) and the question is whether there is a variable assignment such that not all literals of a clause share the same truth value. This problem is NP-hard by a reduction from the problem NAE3SAT: We replace each negated variable $\overline{x_i}$ by a new variable y_i and append the clause $y_i \vee x_i \vee x_i$.

Let ϕ be a monotone 3CNF formula with n variables $v_1, \dots, v_n$ and m clauses $C_1, \dots, C_m$. We construct an $(n+3m+l-2) \times (3m+l-2)$ incomplete haplotype matrix B. The first n rows, which we call *variable rows*, are identified with the variables of ϕ. The next $3m$ rows and the first $3m$ columns are called *literal rows* and *literal columns*, respectively. The remaining $l-2$ columns are marked by $b_1 \dots b_{l-2}$. First, we describe the non-?-entries of the upper left $(n+3m) \times (3m)$ submatrix: Let C_j be a clause of ϕ with literals $\{l_j^1, l_j^2, l_j^3\}$. For each literal l_j^k and its corresponding variable v_i, we put a 1 entry in row v_i and column l_j^k. Then we put the submatrix $\left[\begin{smallmatrix} 1 & 0 & ? \\ ? & 1 & 0 \\ 0 & ? & 1 \end{smallmatrix}\right]$ in columns l_j^1, l_j^2 and l_j^3 and rows l_j^1, l_j^2 and l_j^3. Finally, we set the lower right $(l-2) \times (l-2)$ submatrix to the identity matrix $\left[\begin{smallmatrix} 1 & 0 & \cdot & 0 \\ 0 & 1 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & 1 \end{smallmatrix}\right]$ and all entries in the upper right $(n+3m) \times (l-2)$ submatrix and the lower left $(l-2) \times 3m$ submatrix to 0. An example for this construction for $l = 6$ is depicted in Figure 1.

We claim that $\phi \in$ MONOTONE NAE3SAT if, and only if, $B \in \text{IDPP}_{\text{leafs}\leq l}$.

First, let B' be a completion of B that admits a directed perfect phylogeny T with at most l leafs. The phylogeny T must satisfy some necessary properties: Consider a column b_i. No other column in B can lie on the same root-to-leaf path as b_i because they contain the submatrix $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$. Thus, the

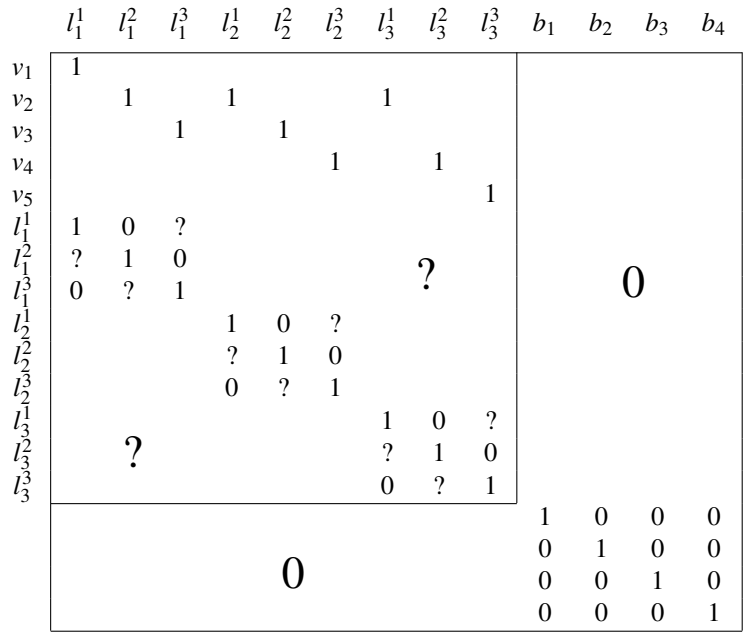

Figure 1: This figure shows the reduced incomplete binary matrix for IDPP$_{\text{leafs}\leq 6}$ and the monotone 3CNF formula $\phi = C_1 \wedge C_2 \wedge C_3$ with clauses $C_1 = v_1 \vee v_2 \vee v_3$, $C_2 = v_2 \vee v_3 \vee v_4$ and $C_3 = v_2 \vee v_4 \vee v_5$. In the matrix, l_j^i denotes the ith literal of clause C_j.

columns $b_1, \ldots, b_{l-2}$ enforce $l-2$ branches $T_1, \ldots, T_{l-2}$ that emanate from the root and a branch T_i contains exactly the column b_i. The remaining columns, which are the literal columns, are distributed to at most two further branches T_{l-1} and T_l since $l-2$ of the permitted leafs are used by $T_1, \ldots, T_{l-2}$. The branches T_{l-1} and T_l are both nonempty because B contains the submatrix $\left[\begin{smallmatrix} 1 & 0 & ? \\ ? & 1 & 0 \\ 0 & ? & 1 \end{smallmatrix}\right]$ in literal columns and each completion of this matrix contains the submatrix $\left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$, which forces columns to lie in different branches. Altogether we obtain that T_{l-1} and T_l are paths. The first n rows assure that all literal columns that correspond to the same variable lie on the same path since they contain 1-entries in the same row. The next $3m$ rows force that the literal columns of each clause do not lie on a single root to leaf path. From this knowledge we construct an assignment τ of the variables of ϕ that witnesses $\phi \in$ MONOTONE NAE3SAT: For a variable v_i, we set $\tau(v_i) = 0$, if the corresponding literal columns lie in T_{l-1} and $\tau(v_i) = 1$, if they lie in T_l.

For the other direction let $\tau : \{v_1, \ldots, v_n\} \to \{0,1\}$ be an assignment of the variables of ϕ such that the literals of each clause do not share the same truth value. We describe, simultaneously, a completion B' for B and a directed perfect phylogeny T with at most l leafs for B'. First we construct $l-2$ branches $T_1, \ldots, T_{l-2}$ from the root. Each branch T_i contains exactly the column b_i and its leaf is labeled with one of the lower $l-2$ rows of B. Two further branches T_{l-1} and T_l (which are paths) are made up by the literal columns and contain completions for the first $n+3m$ rows. The path T_{l-1} contains all literal columns whose corresponding variables are set to 0 by τ in root to leaf order as follows: Consider the clause C_1 with literals $\{l_1^1, l_1^2, l_1^3\}$. If exactly one literal l_1^k of C_1 equals 0, we append a new edge to T_{l-1} that is labeled with the literal column l_1^k. The node after this edge is labeled by the literal row l_1^k. This positioning yields a completion for this row. If exactly two literals l_1^k and l_1^p of C_1 are set to 0 and $p \equiv k+1 \pmod 3$, we append two new edges. We label the first edge with the literal column l_1^k and the subsequent node with the literal row l_1^k. The second edge is labeled with column l_1^p and the node with row l_1^p. Again, we obtain a completion for the rows. Similarly we proceed for the remaining clauses $C_2, \ldots, C_m$ in increasing order. The last node of T_{l-1} is labeled by all variable rows that are set to 0,

which constitutes a completion for these rows. We construct the branch T_l analogously by using rows and columns that correspond to variables that are set to 1. Thus, we obtain a completion of B and a directed perfect phylogeny with at most l leafs. □

The theorem that we just proved has a number of consequences. As mentioned in the introduction, $\text{IDPP}_{\text{leafs}\leq l}$ can easily be reduced to many other problems, including $\text{IPP}_{\text{leafs}\leq l}$, $\text{IDPPH}_{\text{leafs}\leq l}$, and $\text{IPPH}_{\text{leafs}\leq l}$ with $l \geq 2$, and, thus, all of these problems are NP-complete, too.

Having a closer look at the matrix constructed in the proof, we see that for $l = 2$ we can prove a slightly stronger result: $\text{IDPP}_{\text{leafs}\leq 2}$ *with at most 4 known entries per row and 3 known entries per column is* NP*-complete.* This is true since MONOTONE NAE3SAT, which is used for the reduction in the proof of Theorem 1.1, is also NP-hard if we restrict to formulas where each variable appears at most 4 times.

Even though this result is admittedly rather specialized, it shows that neither the maximum number of non-?-entries per row nor per column are parameters for which $\text{IDPP}_{\text{leafs}\leq 2}$ can be fixed parameter tractable. Indeed, for either of these parameters $\text{IDPP}_{\text{leafs}\leq 2}$ does not lie in XP, unless P = NP. This shows that the same it true for IPPH and when we wish to prove fixed-parameter results for IPPH, as we do in the following section, we must look at other parameters.

3 Fixed Parameter Tractability Results

The NP-hardness of $\text{IDPP}_{\text{leafs}\leq l}$ proved in Section 2.2 directly transfers to $\text{IPPH}_{\text{leafs}\leq l}$ using trivial reductions. We also saw that, at least in the case $l = 2$, the problems stay hard when we fix the number of known entries allowed in every column of the input matrix. However, bounding the number of allowed missing entries per column by a constant k makes the problems tractable. We show even more: the problems are fixed-parameter tractable with respect to this parameter. For an introduction on fixed-parameter algorithms and their applications in phylogenetics see [12]. In [13] Gramm et al. showed fixed-parameter tractability for $\text{IDPPH}_{\text{leafs}\leq 2}$. Their result is generalized by our second main result, Theorem 1.2 from the introduction. To prove it, we focus on proving the following, slightly weaker, but more accessible theorem and our proof will implicitly yield the main theorem also:

Theorem 3.1. *Let $l \geq 2$. Then $\text{IPPH}_{\text{leafs}\leq l}$ is fixed-parameter tractable with respect to the maximal number of ?-entries per column.*

To prove this result we present a fixed-parameter algorithm for $\text{IPPH}_{\text{leafs}\leq l}$, that is an algorithm running in time $f(k) \cdot n^{O(1)}$, where k is the maximal number of missing entries per columns, f is some function, and n is the length of the input. Note that the number l of allowed leafs is not treated as a parameter but as a constant that is fix in the problem.

We present this fixed-parameter algorithm in Sections 3.1 to 3.3. However, although it ensures fixed-parameter tractability, we will see that it cannot be considered as efficient, because the function $f(k)$ will be in the range of $3^{k!}$. In Section 3.4 we present a more efficient fixed-parameter algorithm for $\text{IDPP}_{\text{leafs}\leq 2}$.

3.1 Plan of the FPT-Algorithm for IPPH with a Bounded Number of Leafs

The structure of the algorithm we present is similar to the fixed-parameter algorithm for IDPPH from Gramm et al. [13]: First, in a preprocessing phase, we modify the input matrix to make sure the number of columns the actual algorithm has to deal with simultaneously is reduced to a number independent of the size of the input. Then we use dynamic programming to construct a perfect phylogeny matching the required restrictions. This will happen by constructing all possible perfect phylogenies column by column, starting at the leafs of the resulting phylogenetic tree and working towards its center.

In this approach it is essential to know which columns to place on which level of the tree, that is, at which distance to a leaf or to the center. For directed phylogenies the leaf count introduced by Gusfield [16] is a good indicator: the leaf count of a column, being the number of 2-entries plus twice the number of 1-entries, counts the number of explaining haplotypes that label nodes in the subtree connected to the column. So a small leaf count indicates a small distance to a leaf and a big leaf count a small distance to the root. In particular the leaf count of columns does not decrease on any path from any leaf to the root. In the undirected case the leaf count looses this helpful quality and since we do not have a distinguished root it is not clear how to measure the distance to the center of the phylogeny. To tackle these problems we generalize the leaf count:

Let c be a column of a genotype matrix A and n_x its number of x-entries for $x \in \{0,1,2\}$. Let T be a perfect phylogeny admitted by A. By deleting the edge labeled c we partition T in two subtrees T_0 and T_1, such that all haplotypes labeling nodes in T_0 have a 0-entry and all haplotypes labeling T_1 have a 1-entry in column c. We name these subtrees *heavy* and *light sides* of c according to the following criteria:

- If $n_0 > n_1$, we call T_0 the heavy side and T_1 the light side of c.
- If $n_0 < n_1$, we call T_1 the heavy side and T_0 the light side of c.
- For $n_0 = n_1$, if the first non-2 entry in c is a 0-entry, we call T_0 the heavy side and T_1 the light side of c, otherwise call T_1 the heavy side and T_0 the light side of c.

We define $hs(c) := n_2 + 2\max\{n_1, n_0\}$ to be the *heavy side size* and $ls(c) := n_2 + 2\min\{n_1, n_0\}$ to be the *light side size* of c. Intuitively, the light side of c is the subtree that has the smaller number of explaining haplotypes for A which are counted by $ls(c)$. Analogously $hs(c)$ counts the number of haplotypes on the heavy side. As for the leaf count, haplotypes that explain multiple genotypes in A are counted a multiple number of times. Note that $hs(c)$ and $ls(c)$ do not depend on T, but only on the genotype matrix A. The *light side value* of c, $lv(c)$, is the column vector containing the values the haplotypes labeling the nodes in the light side of c have in column c, the *heavy side value*, $hv(c)$, is defined analogously.

The light side size will be our criterion how to handle columns. When constructing a phylogeny T, columns with small light side size will be handled early and placed in the outer regions of T. Also this notation allows us to specify a center by distinguishing a root. Before we state this more precisely in Lemma 3.2, we define a pre-order $\leq$ that serves the same purpose for undirected perfect phylogenies as does $\preceq$ for directed perfect phylogenies.

Let c and d be $\{0,1,2\}$-columns of length n. We write $c \leq d$ if, and only if, for every $i \in \{1,\ldots,n\}$ the following holds: $d[i] = hv(d)$ implies $c[i] = hv(c)$, and $c[i] = lv(c)$ implies $d[i] = lv(d)$. If $c \leq d$ and $d \leq c$, we say that c and d are *equivalent* and write $c \equiv d$. Note that if c has 0-entries exactly in the places were d has 1-entries, and 1-entries exactly in the places where d has 0-entries, then $c \equiv d$, so we consider two columns that can be obtained from each other by flipping 0- and 1-entries as equivalent. This is justified by the fact that if a haplotype matrix B admits a perfect phylogeny, then B also admits a perfect phylogeny in which all columns that are equivalent label the same edge. Note that for a set of columns C that does not contain two equivalent columns $\leq$ is a partial order on C. We call $\leq$ the *weight order*. It is easy to see that the weight order is extended by the light side order, that is, $c \leq d$ implies $ls(c) \leq ls(d)$.

From now on we consider only genotype matrices that have no two distinct columns c and d such that $c \equiv d$.

Lemma 3.2. *Let A be a genotype matrix that admits a perfect phylogeny T. Then there is exactly one node $r \in V$ that is in the heavy side of every column of A. Furthermore, the columns that label a path from any node to r increase with respect to the weight order.*

Proof. Let B be the explaining haplotype matrix admitting T. First we show the existence of a node in the heavy side of all columns. Let c be a column with maximal light side size in A and let r be the node

from the heavy side of c that is adjacent to c. We prove that r also is in the heavy side of every other column from A. Let $d \neq c$ be a column from A. Note that d cannot label the same edge of c, nor can B contain no haplotype that labels a node between d and c, because both would imply $c \equiv d$ which we presume not to be the case. That means d either lies in the light or in the heavy side of c. If d lies in the light side, then one of its sides contains the heavy side of c and the other side is contained in the light side if c. Since $c \not\equiv d$ both inclusions are strict and the size of the side containing the heavy side of c is greater than $hs(c)$, which means that this side is the heavy side of d. Hence, r is in the heavy side of d. If d is in the heavy side of c, then analogously one of its sides includes the light side of c and its size is greater than $ls(c)$. Since the c has maximal light side size, this is the heavy side of d and it contains r, because r is on the path from d to c. Thus r is in the heavy side of every column from A.

It is easy to see that r is unique: let s be a node different from r and let c be a column on the path from r to s, then, since r is in the heavy side of c, s is in its light side.

We proceed with showing that the edges in every path to r are ordered increasingly. Let v be some node in T. We show that the weight order is a linear order on the edges belonging to a path from v to r. Let c and d be columns that label the path, such that the edge labeled c comes first on the way from v to r. (Again, note that c and d cannot label the same edge, because $d \not\equiv c$.) Since r is in the heavy side of both columns, the light side of c is a subset of the light side of d, that means every explaining haplotype having a $lv(c)$-entry in column c also has a $lv(d)$-entry in column d and every haplotype having a $hv(d)$-entry in column d, also has $hv(c)$-entry in column c. Hence, $c < d$, which completes the proof. □

It follows directly that in every perfect phylogeny the columns in a path to the vertex r from the previous lemma have increasing light side sizes. For a genotype matrix A we denote the haplotype that has in every column c a $hv(c)$-entry by $root(A)$.

The concept of heavy and light sides enables us to transfer methods from the directed to the undirected case. Informally speaking, a 1-entry in a directed problem instance corresponds to a light side value, the all-0-haplotype corresponds the haplotype $root(A)$, and the leaf count corresponds to the light side size.

3.2 The Dynamic Program

The dynamic program will be the main routine of the FPT-algorithm for $\text{IPPH}_{\text{leafs}\leq l}$. Given an incomplete genotype matrix A it will for a light side size look at all columns that can be completed to match this size and merge them into phylogenies build previously from columns with smaller light side sizes. That means we construct phylogenies from outside to inside inserting column per column in the order of their light side sizes.

The following lemma contains a characterization that makes it easy to insert single columns into previously build (partial) phylogenies. We call two columns *compatible* if in every place were one of them has its light side value, the other one has its heavy side value. An arbitrary number of columns are compatible if every two of them are compatible.

Lemma 3.3. *Let A be a genotype matrix. And let T be a tree in which every edge is labeled by exactly one column of A and every column labels exactly one edge. Then the nodes of T can be labeled such that T is a perfect phylogeny admitted by A if and only if the following conditions are satisfied:*

1. *there is a node r such that all paths from any node to r are ordered increasingly with respect to the weight order* (ordering condition)*,*
2. *for every node v all columns that are adjacent to v and do not belong to the path from v to r are compatible* (compatibility condition)*,*
3. *for every node v and every genotype $g \in A$ there are no three columns c_1, c_2, c_3 adjacent to v such that $g[c_1] = g[c_2] = g[c_3] = 2$* (2-path condition)*.*

Proof. First let T be a perfect phylogeny admitted by A. We show that the three conditions are satisfied: The ordering condition holds due to Lemma 3.2. Now assume Condition 2 is not satisfied at some node v of T. That means there are two columns c, c' adjacent to v and not on the path to r and there is a genotype $g \in A$, such that $g[c] = lv(c)$ and $g[c'] = lv(c')$. All haplotypes explaining g must have the same entries in these columns, so the must belong to the light sides of both c and d, which is impossible because those are disjoint. Thus, the compatibility condition is satisfied.

For the 2-path condition let v be an edge. Assume there are three adjacent edges labeled by columns c_1, c_2, c_3 and a genotype $g \in A$ such that $g[c_1] = g[c_2] = g[c_3] = 2$. Let h and h' the haplotypes explaining g in T. For each column from $\{c_1, c_2, c_3\}$, one of h and h' must belong to its light side and the other one to its heavy side. Since only one of the columns adjacent to v can belong to the path from v to r, we can assume without loss of generality that c_1 and c_2 do not belong to that path. Then c_1 and c_2 have disjoint light sides. So we can assume h to label a node in the light side of c_1 and h' to label a node in the light side of c_2. If c_3 also does not lie on the bath from v to r, then none of h and h' belong to the light side of c_3 because it is disjoint to either light sides of c_1 and c_2. If c_3 belongs to that path, then both h' and h lie in the light side of c_3. So we have a contradiction and thus, all three conditions hold.

To prove the other direction let T satisfy the three conditions. We show that the labeling induced by assigning the haplotype having in every column its heavy side value to r makes T a perfect phylogeny admitted by A. For this it suffices to show that for every genotype $g \in A$ there are to labels explaining it. Let g be a genotype from A and let $C^{lv} := \{c \in C \mid g[c] = lv(c)\}$ and $C^2 := \{c \in C \mid g[c] = 2\}$ be the sets of columns in which g has a light side value, or respectively a 2-entry. The rest of the proof is organized as follows: we show that

(a) there is a node v that is connected to r via a path that uses exactly the columns from C^{lv},
(b) there are two nodes w, w' connected by a path that uses exactly the columns from C^2 and goes through v,
(c) the labels of w and w' explain g.

Because T is ordered the columns from C^{lv} form a connected component in T that contains r. The compatibility condition ensures that this component is a single path from r to some node v. Thus (a) is true. Let T_v be the subtree rooted at v. Assume there is a $c \in C^2$ that labels an edge not belonging to T_v. Let v' be the latest ancestor node of both columns and let d_1 and d_2 be the columns connected to v' such that d_1 starts the path from v' to v and d_2 starts the path from v' to c. Because of the ordering condition g has a $lv(d_1)$-entry in d_1 and a 2 or a lv-entry in d_2. This contradicts the compatibility condition in v'. So, all columns from C^2 are in T_v. Because of the ordering condition, C^2 forms a connected component in T_v that contains v. The 2-path condition implies that this component is a path, so (b) holds. Let w, w' be the two vertices that are connected by this C^2-path. The paths from r to w and from r to w' contain all columns from C^{lv}, so the labels have a light side value in these columns. Every column from $c \in C^2$ belongs to exactly one of those paths, therefore one of both labels has $lv(c)$ and the other $hv(c)$ in column c. All columns not belonging to C^{lv} or C^2 do not appear in either path, thus all other columns contain heavy side values. Hence, the labels of w and w' explain g. □

Note that compatibility and 2-path condition can be verified locally for each vertex. Let A be a genotype matrix with column set C and let $D \subseteq C$. We identify D with the submatrix of A that has column set D and talk about perfect phylogenies for D, $root(D)$, etc. Let T be a perfect phylogeny admitted by D, then the *center* of T, $center(T)$, is the set $\{c \in D \mid c \text{ is adjacent to the vertex labeled by } root(D) \text{ in } T\}$. We show that, to check whether a new column can be inserted in a previously constructed phylogeny T, it is sufficient to consider the center of T. For sets of columns $C \subseteq C'$ admitting perfect phylogenies T and T' we say that T' *extends* T if for all $c_1, c_2 \in D$ it holds: c_1 lies on the path from c_2 the vertex labeled $root(C)$ in T if and only if c_1 lies on the path from c_2 to the vertex labeled $root(C')$ in T'. Intuitively this means that T' can be obtained from T by splitting up vertices and connecting them with new edges. In the following we denote by $l(T)$ the number of leafs of a perfect phylogeny T.

Lemma 3.4. *Let A be a genotype matrix and C, D subsets of its column set such that C admits a perfect phylogeny T and $c \in C, d \in D$ implies $ls(c) \leq ls(d)$. Let $L \in \mathbb{N}$ and $H \subseteq C$. Then the following is equivalent:*

- *$C \cup D$ admits perfect phylogeny U with L leafs, such that U extends T and $center(U) = H$,*
- *$center(T) \cup D$ admits perfect phylogeny U with $L - l(T) + |center(T)|$ leafs, such that every column from $center(T)$ is adjacent to a leaf and $center(U) = H$.*

Proof. First let $C \cup D$ admit a perfect phylogeny U with L leafs, such that U extends T. Since $center(T) \cup D \subseteq C \cup D$, $center(T) \cup D$ admits a perfect phylogeny U' that can be extended to U. Assume there is a $c \in center(T)$ that is not adjacent to a leaf in U'. Then there is another column $d \in center(T)$ such that the path from the node $root(center(T) \cup C)$ to d goes through c. Because of the ordering condition $d < c$ and also $ls(d) < ls(c)$, therefore $d \notin D$. On the other hand if $d \in center(T)$, then it holds as well for U and T that c lies on the path from the root-node to d, because U extends U' and T. This contradicts the fact, that c and d are both adjacent to the root node in T. Hence each $c \in center(T)$ is adjacent to a leaf in U'. Further, it holds that every column from D is adjacent to a leaf in U if and only if it is in U'. Additionally all columns from C that are adjacent to a leaf in U are exactly those adjacent to a leaf in T. In U' all columns from $center(T)$ are adjacent to leafs, therefore U' has exactly $L - l(T) + |center(T)|$ leafs. To see that $center(U') = center(U)$, note that no column from $C \setminus center(T)$ can be in $center(U)$ because U extends T. Since U extends U' and those columns from $C \setminus center(T)$ are the only columns "missing" in U', it follows $center(U') = center(U)$.

For the other direction let $center(T) \cup D$ admit a perfect phylogeny U with $L - l(T) + |center(T)|$ leafs, such that every column from $center(T)$ is adjacent to a leaf and $center(U) = H$. We extend U to U' by replacing each leaf v that is adjacent to a $c \in center(T)$ by the light side of c in T. All vertices in U' have a corresponding vertex in U or in T that has the same set of columns labeling adjacent edges. So all vertices in U satisfy the compatibility condition and the 2-path condition. The ordering condition is satisfied as well because two columns label consecutive edges in a path in U' starting at $root(C \cup \{d\})$ if and only if they label consecutive edges in a path in U starting at $root(center(T) \cup \{d\})$ or in a path in T starting at $root(C)$. Hence, U' is a perfect phylogeny that is admitted by $C \cup D$. Since we did not add any columns to the root node, both columns have the same center. We removed the leafs v_c adjacent to columns $c \in center(T)$ and added all leafs from T, therefore U has exactly L leafs. □

Before presenting the algorithm we need some terminology. A *(partial) completion* of an incomplete genotype column c is a column c' obtained by filling (some of) the missing entries with 0, 1, or 2. A *quasi-completion* of c is a column c' that is equivalent to some completion of c. Let C be a set of incomplete genotype columns. A *quasi-completion* of C is a set containing a quasi-completion of each $c \in C$. We also require that in a quasi-completion there are no two columns c, c' such that $C \equiv c'$. *quasi-completions*(C) denotes the set of all quasi-completions of C. The reason why we use quasi-completions instead of completions is, that, to be able to use the previous results, our algorithm needs to produce completions that contain no equivalent columns. Let D be a quasi-completion of C and D' a quasi-completion of a set of incomplete genotype columns C'. Note that C can be completed to admit a perfect phylogeny if and only if C has a quasi-completion admitting a perfect phylogeny. We define D and D' to be *consistent* if for each $c \in C \cap C'$ there is a quasi-completion of c in $D \cap D'$. In other words, two quasi-completions are consistent if all columns that are completed in both sets are completed in the same way.

Let C be the column set of a genotype matrix A. Note that the light size side of a column ranges between 0 and the number n of genotypes in A. We define $C^{=i} := \{c \in C \mid ls(c) = i\}$, $C^{\leq i} := \{c \in C \mid ls(c) \leq i\}$, $C^{\geq i} := \{c \in C \mid ls(c) \geq i\}$ to be the subsets of columns with light side size equal to i, at most i, at least i respectively.

The input to the dynamic program is an incomplete genotype matrix. Given a column set C of an incomplete genotype matrix A, we use $A_{[i,j]}$ with $0 \leq i \leq j$ to denote the set $\{c \in C \mid i \leq ls(c) \leq j\}$ of

all columns from C with light side size between i and j. So, to distinguish incomplete from complete columns we use subscripts to refer to the light side size of incomplete columns and superscripts if the columns are completed.

The dynamic program fills a table $\mathscr{H}$ as follows: for every light side size i the program considers $A_{[i-2k,i]}$, which is the set of all columns that can be completed to have light side size i, and for every quasi-completion R of $A_{[i-2k,i]}$ generates the entry

$$\begin{aligned}\mathscr{H}(R,i) := \{\, (center(T),L) \mid & C \in \textit{quasi-completions}(A_{[0,i]}),\\ & C \text{ consistent with } R,\\ & C^{\leq i} \text{ admits perfect phylogeny } T \text{ with } L \leq l \text{ leafs} \,\}.\end{aligned} \tag{1}$$

Informally $\mathscr{H}(R,i)$ contains information about all perfect phylogenies with up to l leafs that are admitted by completions of $A_{[0,i]}$ consistent with R. Note that entries in column n represent perfect phylogenies for A that have at most l leafs. The table is filled column per column starting with light side size 0 and ending with light side size n. To compute an entry $\mathscr{H}(R,i)$ the program uses the previously generated entries in columns $i-1$ according to the following equation.

Lemma 3.5. *Let A be an incomplete genotype matrix, $1 \leq i \leq n$, and $R \in \textit{quasi-completions}(A_{[i-2k,i]})$. Then*

$$\begin{aligned}\mathscr{H}(R,i) = \{\, (center(T),L) \mid & S \in \textit{quasi-completions}(A_{[i-1-2k,i-1]}),\\ & S \textit{ consistent with } R,\\ & (H,p) \in \mathscr{H}(S,i-1),\\ & H \cup R^{=i} \textit{ admits perfect phylogeny } T \textit{ with } L+|H|-p \textit{ leafs},\\ & \textit{in } T \textit{ every } c \in H \textit{ is adjacent to a leaf} \,\}.\end{aligned}$$

Proof. $\subseteq$: Let $(center(T),l(T)) \in \mathscr{H}(R,i)$. That means there is $C \in \textit{quasi-completions}(A_{[0,i]})$ consistent with R and T is a perfect phylogeny with $L \leq l$ leafs admitted by $C^{\leq i}$. Let D be a subset of C that completes $A_{[0,i-1]}$ and S a subset of D that completes $A_{[i-1-2k,i-1]}$. Then we know that $S \in \textit{quasi-completions}(A_{[i-1-2k,i-1]})$ is consistent with R and D. Let U be the phylogeny admitted by $D^{\leq i-1}$ that can be extended to T. It holds that $(center(U),l(U)) \in \mathscr{H}(S,i-1)$. Since $D^{\leq i-1} \cup R^{=i} = C^{\leq i}$, Lemma 3.4 implies that $center(U) \cup R^{=i}$ admits a perfect phylogeny T' with $l(T)+l(U)-|center(U)|$ leafs in which every column from $center(U)$ is adjacent to a leaf. Thus, $(center(T),l(T))$ belongs to the set on the right hand side.

$\supseteq$: For the other direction, let $(center(T),L)$ be from the set on the right hand side, that means there is an $S \in \textit{quasi-completions}(A_{[i-1-2k.i-1]})$ consistent with R and $(H,p) \in \mathscr{H}(S,i-1)$ such that T is admitted by $H \cup R^{=i}$, has $L+|H|-p$ leafs, and each of its edges labeled by a column from H is adjacent to a leaf. Since $(H,p) \in \mathscr{H}(S,i-1)$, there exists $C \in \textit{quasi-completions}(A_{[0,i-1]})$ that is consistent with S and $C^{\leq i-1}$ admits a perfect phylogeny U with p leafs and $center(U) = H$. According to Lemma 3.4 it follows that $C^{\leq i-1} \cup R^{=i}$ admits a perfect phylogeny T' such that T' extends both T and U, $center(T') = center(T)$, and T' has $l(T)+l(U)-|H| = L$ leafs. It holds that C is consistent with R, therefore $C \cup R \in \textit{quasi-completions}(A_{[0,i]})$ is also consistent with R and $(C \cup R)^{\leq i} = C^{\leq i-1} \cup R^{=i}$. Thus, $(center(T),L) \in \mathscr{H}$. □

Theorem 3.6. *The algorithm* DECIDE-IPPH$_{\text{leafs}\leq l}$ *(depicted in Figure 2) decides whether an incomplete genotype matrix admits a perfect phylogeny with up to l leafs in time $3^{O(k^2\lambda(A))}\lambda(A)^{l+1}nm^l$ where k is the maximal number of missing entries per column and $\lambda(A) = \max_i |A_{[i,i]}|$ is the maximal number of columns from A having equal light side size.*

Algorithm : DECIDE-IPPH$_{\text{leafs}\leq l}$
Input : An $n \times m$ genotype matrix A with at most k missing entries per column.
Output : "yes" if IPPH$_{\text{leafs}\leq l}$, "no" otherwise

1 delete all columns in which all entries are from $\{0,?\}$ or all entries are from $\{1,?\}$
2 $\mathscr{H}(\emptyset,0) \leftarrow \emptyset$
3 **for** $i \leftarrow 1$ to n **do**
4 **for all** $R \in$ *quasi-completions*$(A_{[i-2k,i]})$ **do**
5 $\mathscr{H}(R,i) \leftarrow \emptyset$
6 **for all** $S \in$ *quasi-completions*$(A_{[i-1-2k,i-1]})$ **do**
7 **if** R and S are consistent **then**
8 $\mathscr{H}(R,i) \leftarrow \mathscr{H}(R,i) \cup$
$\{(center(T),L) \mid (H,p) \in \mathscr{H}(S,i-1),$
$H \cup R^{=i}$ admits perfect phylogeny T with $|H|+L-p$ leafs,
in T every $c \in H$ is adjacent to a leaf$\}$
9 **if** there is $R \in$ *quasi-completions*$(A_{[n-2k,n]})$ with $\mathscr{H}(R,n) \neq \emptyset$ **then return** "yes"
10 **else return** "no"

Figure 2: Algorithm DECIDE-IPPH$_{\text{leafs}\leq l}$ solves the IPPH$_{\text{leafs}\leq l}$ problem for arbitrary instances. It is used as a subroutine in the fixed-parameter algorithm.

Proof. Correctness. Since the property of admitting a perfect phylogeny with at most l leafs is invariant under adding or deleting all-0 or all-1 columns, line 1 does not change this property for A. Now, $A_{[0,0]}$ is empty, therefore in line 2 column 0 of $\mathscr{H}$ is filled matching Equation (1). According to Lemma 3.5 all other entries in the table also satisfy Equation (1). Hence, the n-th column of $\mathscr{H}$ has an entry different from $\emptyset$ if and only if there is a quasi-completion of $A_{[0,n]} = A$ that admits a perfect phylogeny with no more than l leafs.

Running time. Processing line 1 can be done in time $O(mn)$. For the running time of the for-loops we start with bounds for sizes of *quasi-completions*$(A_{[i-j,i]})$ and $\mathscr{H}(R,i)$. First note that $A_{[i-j,i]}$ contains at most $(j+1)\lambda(A)$ columns. Since in every column at most k entries need to be completed with values from $\{0,1,2\}$ and for each completion c we can chose between c and its "flipped" version equivalent to C, it holds $|$*quasi-completions*$(A_{[i-j,i]})| \leq 3^{k(j+1)\lambda(A)}2^{(j+1)\lambda(A)}$. The center of a perfect phylogeny with L leafs contains at most L columns because every center columns starts a path from the root to a leaf. Hence, an element of $\mathscr{H}(R,i)$ consists of a set of up to l columns and a number smaller or equal to l. There are at most m^l to chose the columns from A and not more than 3^{kl} ways to complete them, therefore $|\mathscr{H}(R,i)| \leq m^l 3^{kl} l$.

The for-loop staring in line 3 goes though n iterations, the for-loop in line 4 through at most $3^{k(2k+1)\lambda(A)}2^{(2k+1)\lambda(A)}$ iterations. To find the quasi-completions $S \in$ *quasi-completions*$(A_{[i-1-2k,i-1]})$ that are consistent with a given quasi-completion $R \in$ *quasi-completions*$(A_{[i-2k,i]})$ we just have to chose quasi-completions for the columns from $A_{[i-1-2k,i-1-2k]}$, because the quasi-completions for the remaining columns are determined by R. Therefore the for-loop in line 6 iterates over at most $3^{k\lambda(A)}2^{\lambda(A)}$ quasi-completions S. In line 8 the algorithm needs to construct all perfect phylogenies T admitted by the union of centers H from $\mathscr{H}(S,i-1)$ and $R^{=i}$, such that each $c \in H$ labels an edge adjacent to a leaf. Note that $R^{=i} \subseteq center(T)$. To construct all those phylogenies T it is sufficient for every $c \in H$ to choose whether it belongs to a light side of a column, and of which column, from $R^{=i}$ and to check whether the compatibility and the 2-path condition are satisfied in all vertices adjacent to a column from $R^{=i}$. So, for a given H there are at most $|R^{=i}|^l$ perfect phylogenies T that can be verified in time $O(|R^{=i}|n)$. Since $|\mathscr{H}(S,i-1)| \leq m^l 3^{kl} l$ and since $|R^{=i}| \leq |A_{[i-2k,i]}| \leq (2k+1)\lambda(A)$, lines 3-8 can be performed in time $O(n3^{k(2k+1)\lambda(A)}2^{(2k+1)\lambda(A)}3^{k\lambda(A)}2^{\lambda(A)}m^l 3^{kl} l(2k+1)^{l+1}\lambda(A)^{l+1}n)$. This term can be simplified to

$3^{O(k^2\lambda(A))}\lambda(A)^{l+1}n^2m^l$.

For line 9 the program has to look up the entries in the last column of $\mathscr{H}$ which are not more than $|\textit{quasi-completions}(A_{[i-2k,i]})| \leq 3^{k(2k+1)\lambda(A)}2^{(2k+1)\lambda(A)}$.

Thus, the total running time is bounded by $3^{O(k^2\lambda(A))}\lambda(A)^{l+1}n^2m^l$. □

Note that $\lambda(A)$ can be as large as m, therefore DECIDE-IPPH$_{\text{leafs}\leq l}$ is not the desired fixed-parameter algorithm. However, in the next section we show that A can be modified such that $\lambda(A)$ has a bound that does not depend on n and m but only on the parameter k.

3.3 The Preprocessing Phase

The aim of the preprocessing routine is to reduce the number of columns with the same light side size. The idea is to find sets of columns that must be completed in the same way to admit a perfect phylogeny with at most l leafs and to replace this columns by a single column allowing the same completions.

First we give a criterion, saying that certain submatrices do not occur in genotype matrices admitting perfect phylogenies with up to l leafs:

Lemma 3.7. *Let $x, y_1, \ldots, y_{l+1} \in \{0,1,2\}$ and $x \neq y_i$ for all i. Then the matrix $A = \begin{bmatrix} y_1 & x & \cdots & x \\ x & y_2 & \cdots & x \\ \vdots & & \ddots & \vdots \\ x & x & \cdots & y_{l+1} \end{bmatrix}$ does not admit a perfect phylogeny with at most l leafs.*

Proof. First assume $x = 2$. Then $\textit{root}(A) = (y_1, \ldots, y_{l+1})$. According to Lemma 3.2 $\textit{root}(A)$ labels a vertex r in each perfect phylogeny for A and the columns on each path starting at r are ordered decreasingly with respect to the weight order. Since all columns from A are incomparable with respect to this order, every perfect phylogeny admitted by A has at least $l+1$ leafs.

Now assume $x \neq 2$. Without loss of generality let $x = 0$. In this case $\textit{root}(A) = (x, \ldots, x)$ and again no two columns of A are comparable, so with the same arguments as above all perfect phylogenies admitted by A have at least $l+1$ leafs. □

A *consensus* c of a set C of incomplete genotype columns is a partial completion of all $c \in C$. The following lemma shows that $\kappa(h) := (3l)^h h!$ columns that have a consensus with $k-h$ missing entries can be replaced by its consensus without altering membership in IPPH$_{\text{leafs}\leq l}$. For the case $l = 2$ this result was shown in [13], the given proof generalizes directly to this version for arbitrary l.

Lemma 3.8. *Let A be a genotype matrix with at most k missing entries per column. Let $h \geq 0$ be the minimal number such that there is a subset C of columns from A with $|C| > \kappa(h)$ that has a consensus c with exactly $k-h$ missing entries. Let A' be the incomplete genotype matrix obtained from A by deleting all columns from C and adding c. Then $A \in$ IPPH$_{\text{leafs}\leq l}$ if and only if $A' \in$ IPPH$_{\text{leafs}\leq l}$.*

Proof. Clearly, $A' \in$ IPPH$_{\text{leafs}\leq l}$ implies $A \in$ IPPH$_{\text{leafs}\leq l}$. For the reverse direction suppose that A can be completed to a genotype matrix E admitting a perfect phylogeny with at most l leafs. Let C' be the set of columns from E that complete the columns from C. We prove that also A' can be completed to admit a perfect phylogeny with up to l leafs. For $h = 0$, we have that $|C| > 1$ and all columns from C are equal to c. So obviously, $A' = A$.

Now we prove the claim for $h > 0$. If there is a column $d \in C'$ that is a completion of c, then E completes A' and therefore $A' \in$ IPPH$_{\text{leafs}\leq l}$. In the rest of the proof, we show that the other case, where there is no such column in C' can not occur. So, for sake of contradiction, suppose that no column from C' is consenting with c. Then each $d \in C$ has at least one ?-entry that faces a non-question mark entry in c and that is completed differently in C' than in c. We call these the disputed positions of d.

The minimality of h implies that every column with $k-h'$ missing entries has dimension at most $\kappa(h')$ for all $h' < h$. 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 $\kappa(h-1)$ missing entries. Indeed, if this were not the case, then the columns of C containing these ?-entries would have a consensus c' having at least one more ?-entry than c does and dimension larger than $\kappa(h-1)$ contradicting the minimality of h.

We form sets C_x for $x \in \{0,1,2\}$ containing those columns in C that have a disputed position facing an x-entry in c. Since every column in C has a disputed position, we have $C = C_0 \cup C_1 \cup C_2$. We show that the cardinality of each of the sets C_0, C_1, and C_2 is not larger than $l \cdot h \cdot \kappa(h-1)$, which implies that C contains at most $\kappa(h)$ columns, a contradiction.

We construct a graph G_x whose vertex set is exactly C_x. Let there be an edge between a vertex $d \in C_x$ and $e \in C_x$ if there is a row in which both d and e have a disputed position that faces an x-entry in c. We claim that the maximum degree of G_x is less than $h \cdot \kappa(h-1)$. To see this, first note that every vertex (which is a column) has at most h disputed positions. Next, for each row there are at most $\kappa(h-1)$ columns having a disputed positions in this row, because otherwise the column obtained from c by replacing the entry in this row by a question mark has dimension larger than $\kappa(h-1)$ contradicting the minimality of h. Thus, each vertex can be adjacent to at most $h \cdot (\kappa(h-1)-1)$ different vertices.

We claim that the largest independent set in G_x has size at most l. To see this, assume that there are $l+1$ independent columns in G_x. Since each of those columns has a disputed position in a row where the other l columns have an x-entry the $(l+1) \times (l+1)$-matrix $\begin{bmatrix} ? & x & \cdots & x \\ x & ? & \cdots & x \\ \vdots & & \ddots & \vdots \\ x & x & \cdots & ? \end{bmatrix}$ is a submatrix of A. Since all the question marks are at disputed positions, they are completed differently from 0 in C'. According to Lemma 3.7, E does not admit a perfect phylogeny with at most l leafs. This is a contradiction.

We just have shown that A_x has maximum degree $h \cdot (\kappa(l-1)-1)$ and independence number at most l. So A_x has at most $l + l \cdot h \cdot (\kappa(h-1)-1) \leq l \cdot h \cdot (\kappa(h-1))$ vertices. Therefore, $|C_x| \leq l \cdot h \cdot (\kappa(h-1))$ which completes the proof. □

The procedure PREPROCESS$_{\text{leafs}\leq l}$ replaces columns according to Lemma 3.8 to obtain an incomplete genotype matrix whose maximum number of columns with the same light side size is bounded by a number that depends only on the parameters k and l. This bound is $\lambda(k) := (2k+1)l\kappa(k)$, where k is the maximum number of missing entries in any column.

Lemma 3.9. *Let A be a genotype matrix with at most k missing entries per column and let A' be the genotype matrix that is obtained by applying the preprocessing procedure* PREPROCESS$_{\text{leafs}\leq l}$ *to A. Then the following holds:*

1. *A has a perfect phylogeny with at most l leafs iff A' has one.*
2. *If A' contains more than $\lambda(k)$ columns with the same light side size, then $A' \notin$ IPPH$_{\text{leafs}\leq l}$.*

Proof. Since whenever the algorithm replaces a set of columns C by a column p the prerequisites of Lemma 3.8 are met, it holds that A can be completed to admit a perfect phylogeny with at most l leafs if and only if A' can.

For some $i \in \{0,\ldots,n\}$ let D be the set of columns d from A' that have light side size i and suppose $|D| > \lambda(k) = (2k+1)l\kappa(k)$. We prove that A' can not be completed to admit a perfect phylogeny with up to l leafs. Since no more than $\kappa(k)$ columns of A' have a consensus, in any completion A' the column from D must be completed in more than $(2k+1)l$ different ways. Every completion of $d \in D$ has light side size between i and $i+2k$. Therefore in each completion of A' there are at least $l+1$ columns with the same light side size. Since in every perfect phylogeny with at most l leafs there can be at most l columns with the same light side size, it follows that $A' \notin$ IPPH$_{\text{leafs}\leq l}$. □

We analyze the running time of PREPROCESS$_{\text{leafs}\leq l}$. In each but the last iteration of the repeat-loop the number of columns decreases by at least 1 since some of the columns are replaced by a consensus.

Procedure : PREPROCESS$_{\text{leafs}\leq l}$
Input : An $n \times m$ genotype matrix A with at most k missing entries per column.
Output : A preprocessed matrix

```
1  repeat
2      for h ← 0 to k do
3          for all columns c of A do
4              for all partial completions p of c with exactly k − l missing entries do
5                  C ← {c' | c' is column of A and partial completion of p}
6                  if |C| > κ(l) then A ← A with all columns in C replaced by p and break
7  until A remains unchanged
```

Algorithm : FPT-IPPH$_{\text{leafs}\leq l}$
Input : An $n \times m$ genotype matrix A with at most k missing entries per column.
Output : "yes" if IPPH$_{\text{leafs}\leq l}$, "no" otherwise.

```
1  call PREPROCESS_{leafs≤l}(A)
2  for i ← 0 to n do
3      if |A_[i,i]| > λ(k) then return "no"
4  return DECIDE-IPPH_{leafs≤l}(A)
```

Figure 3: Algorithm FPT-IPPH$_{\text{leafs}\leq l}$ is a fixed-parameter algorithm for IPPH$_{\text{leafs}\leq l}$. The parameter is the maximal number of missing entries per column. It uses the PREPROCESS$_{\text{leafs}\leq l}$ procedure to reduce the number of columns with the same light side size.

Thus, there can be at most m iterations of the repeat-loop. The three inner loops iterate over at most $km4^k$ candidates and computing the set C for a given column p can be done in time $O(nm)$. Hence, the algorithm terminates in time $O(k4^k m^3 n)$.

Now we can combine the preprocessing and the dynamic program to obtain a fixed-parameter algorithm for IPPH$_{\text{leafs}\leq l}$.

Proof of Theorem 3.1. We show that FPT-IPPH$_{\text{leafs}\leq l}$ is a fixed-parameter algorithm for IPPH$_{\text{leafs}\leq l}$ where the fixed parameter is the maximal number of missing entries per column.

The correctness follows from Lemma 3.9 and Theorem 3.6. Now we analyze the running time: calling PREPROCESS$_{\text{leafs}\leq l}$ the takes time $O(k4^k m^3 n)$. The for-loop can be processed in time $O(n^2 m)$. After the preprocessing A has at most $\lambda(k)$ columns with the same light side size. Therefore the dynamic program DECIDE-IPPH$_{\text{leafs}\leq l}$ runs in time $3^{O(k^2\lambda(k))}\lambda(k)^{l+1}n^2m^l$. This is the dominating factor and can be simplified to time $3^{O(k^2(3l)^k k!)}n^2m^l$. Thus, IPPH$_{\text{leafs}\leq l}$ is fixed-parameter tractable with respect to the maximal number of ?-entries per column. □

3.4 FPT Algorithms for Perfect Path Phylogeny Problems

In this section we present specialized algorithms for the problems IDPP$_{\text{leafs}\leq 2}$ and IPP$_{\text{leafs}\leq 2}$. First we prove the following result:

Theorem 3.10. IDPP$_{\text{leafs}\leq 2}$ *and* IPP$_{\text{leafs}\leq 2}$ *are fixed-parameter tractable with respect to the maximal number of missing entries per row and*

Note that these problems are also fixed-parameter tractability with respect to the maximal number of missing entries per column. For IDPP$_{\text{leafs}\leq 2}$, this follows from results in [13] and, for IPP$_{\text{leafs}\leq 2}$, this follows from Theorem 3.1.

Beside these, we present a simple fixed-parameter algorithm for $\text{IDPP}_{\text{leafs}\leq 2}$ where the parameter is the maximal number of missing entries per row plus the maximal number of missing entries per column. While this parameter normally exceeds the above ones, our algorithm can be easily implement and the exponential fraction as well as the polynomial fraction of the time bound are lower than in the previous section.

To prove Theorem 3.10, we first present forbidden submatrix characterizations of haplotype matrices that admit (directed) perfect phylogenies with an bounded number of leafs. Similar characterizations are also known for general perfect phylogenies: Haplotype matrices that admit perfect phylogenies and directed perfect phylogenies are characterized by the absence of the submatrices $\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$ and $\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$, respectively. Independent from the objective of this section, forbidden submatrix characterizations give an interesting insight into phylogenetic problems. The forbidden submatrix characterization of perfect phylogenies with at most l leafs is as follows:

Lemma 3.11. *A haplotype matrix B admits a perfect phylogeny with at most l leafs if, and only if, B does not contain the submatrix $F_1 := \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$ and none of the matrices that arise by inversions of columns from the $(l+1) \times (l+1)$ identity matrix $F_2 := \begin{bmatrix} 1 & 0 & \cdot & 0 \\ 0 & 1 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & 1 \end{bmatrix}$.*

Proof. The "only if"-part follows directly from Lemma 3.7 and the fact that haplotype matrices containing F_1 as a submatrix do not admit a perfect phylogeny.

For the "if"-part let B be a haplotype matrix that does not admit a perfect phylogeny with at most l leafs. Since all-0 or all-1 columns have no impact on the number of leafs needed, we assume B to have none of them. If B does not admit a perfect phylogeny at all, it contains the submatrix F_1. Therefore assume B admits a perfect phylogeny T with more than l leafs. For a leaf i, let c_i be a column on its incident edge. Because all columns contain both 0-entries and 1-entries, each leaf i is labeled with a haplotype h_i from B. With the definition of perfect phylogenies follows that $h_i[c_i] = a \in \{0,1\}$ implies $h_j[c_i] = 1-a$ for all $j \neq i$. Thus, F_2, possibly with inverted columns, is a submatrix of B. □

The following lemma states a characterization of directed perfect phylogenies with at most l leafs:

Lemma 3.12. *A haplotype matrix B admits a directed perfect phylogeny with at most l leafs if, and only if, B does not contain one of the following submatrices: $G_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$, the $(l+1) \times (l+1)$ identity matrix $F_2 = \begin{bmatrix} 1 & 0 & \cdot & 0 \\ 0 & 1 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & 1 \end{bmatrix}$ and the $l \times (l+1)$ matrix $G_2 = \begin{bmatrix} 1 & 0 & \cdot & 0 & 1 \\ 0 & 1 & \cdot & 0 & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdot & 1 & 1 \end{bmatrix}$*

Proof. We show that a haplotype matrix B contains one of the submatrices G_1, G_2 and F_2 if and only if B' obtained from B by adding the all-0-row contains one of the forbidden matrices of Lemma 3.12.

The "only if"-part is obvious. We show the "if"-part. If B' contains F_1, then B contains G_1. If B' contains F_2, then also B does. If B' contains F_2 with one inverted column, then B contains G_2 and, finally, if B' contains F_2 with at least two inverted columns then B, contains G_1.

□

For $l = 2$, the submatrix condition of Lemma 3.12 still holds if we transpose a given haplotype matrix, which implies the following corollary:

Corollary 3.13. *A haplotype matrix B admits a directed perfect path phylogeny if, and only if, its transposition B^T admits a directed perfect path phylogeny.*

Note, that the property of Corollary 3.13 does not hold for undirected perfect phylogenies and directed perfect phylogenies with at least 3 leafs.

Now we are able to prove Theorem 3.10:

Proof of Theorem 3.10. To solve $\text{IDPP}_{\text{leafs}\leq 2}$, we transpose a given incomplete haplotype matrix and use the fixed-parameter algorithm for $\text{IDPPH}_{\text{leafs}\leq 2}$ from [13] to solve it. Because the parameter of this algorithm is the maximal number of missing entries per column, we obtain an algorithm where the parameter is the maximal number of missing entries per row.

The fixed-parameter algorithm for $\text{IPP}_{\text{leafs}\leq 2}$ works as follows: given an incomplete haplotype matrix B, we consider all completions r' of a row r, which leads to partial completions of B. For each r', we direct the matrix B and obtain matrix B'. That means, if $r'[i] = 1$ for a column i, we substitute each 1-entry in column i with a 0-entry and each 0-entry with a 1-entry. For B holds that $B \in \text{IPP}_{\text{leafs}\leq 2}$ if, and only if, $B' \in \text{IDPP}_{\text{leafs}\leq 2}$. Thus, we can use the algorithm for $\text{IDPP}_{\text{leafs}\leq 2}$, which was described above, as a subroutine in the iteration. If the subroutine call returns "yes", we know $B \in \text{IPP}_{\text{leafs}\leq 2}$. If each subroutine call returns "no", we know $B \notin \text{IPP}_{\text{leafs}\leq 2}$. Overall we have an algorithm for $\text{IPP}_{\text{leafs}\leq 2}$ that is parametrized by the maximal number of missing entries per row. □

A Simple FPT Algorithm for IDPP Restricted to Two Leafs. We present a fixed-parameter algorithms for $\text{IDPP}_{\text{leafs}\leq 2}$ where the parameter is the maximal number of missing entries per row plus the maximal number of missing entries per column. The algorithm uses the following graph, which was first defined in [22]: Let B be an incomplete haplotype matrix. The vertex set of the undirected graph G_B^1 is made up by the rows and the columns of B and there is an edge between a row vertex and a column vertex if B contains a 1-entry in the corresponding row and column. The graph G_B^1 represents necessary conditions for directed perfect path phylogenies:

Lemma 3.14. *Let B be a haplotype matrix where each row and column contains a 1-entry and T a directed perfect path phylogeny with branches T_0 and T_1 for B. If two vertices in G_B^1 are connected, then the corresponding rows and columns lie in the same branch of T.*

Proof. Let T be a directed perfect path phylogeny for a haplotype matrix B and c and c' two column vertices that are connected in G_B^1. Let $c, r_1, c_2, \ldots, r_m, c'$ be a path between these vertices where the r_i are row vertices an the c_i are column vertices. Row r_1 contains a 1-entry in column c and lies, therefore, in the same branch with c. Column c_2 contains a 1-entry in row r_1 and lies, therefore, on the path between r_1 and the root. If we use these arguments inductively, we obtain that c and c' lie in the same branch of T. The same holds for row vertices and combinations of row and column vertices. □

Our algorithm seeks for partitions of rows and columns that are described in the following lemma:

Lemma 3.15. *A haplotype matrix B admits a directed perfect path phylogeny if, and only if, there exist a partition of the rows into sets R_0 and R_1 and a partition of the columns into sets C_0 and C_1 such that (a) the submatrices induced by R_0 and C_0 and by R_1 and C_1 admit phylogenetic sortings and (b) the submatrices induced by R_0 and C_1 and by R_1 and C_0 contain only 0-entries.*

Proof. Let T be a directed perfect path phylogeny with branches T_0 and T_1 for the haplotype matrix B. If a row and a column lie in different branches, they do not share a 1-entry. Also each branch is a phylogenetic sorting for the submatrix that is induced by its rows and columns. Thus T_0 and T_1 induce suitable partitions for rows and columns.

The other way round, let C_0, R_0, C_1 and R_1 be as above with induced submatrices B_0 and B_1. By identifying the root nodes of the phylogenetic sortings for B_0 and B_1, we obtain a directed perfect path phylogeny for B. □

The algorithm for IDPP works as follows: Given an incomplete haplotype matrix B, it iterates over all completions of a row r and a column c that share a 1-entry. This yields a partial completion B' of B. For each B' we use the graph $G_{B'}^1$ to compute a partition of the rows into sets R_0 and R_1 and a partition of the columns into sets C_0 and C_1. A row lies in R_0 if it is connected to r and in R_1 otherwise. Similarly we

proceed for the columns. After this the algorithm checks whether the induced submatrices for R_0 and C_0 and for R_1 and C_1 lie in $\text{IDPP}_{\text{sorted}}$. If this is true, the algorithm returns "yes". If this is not possible for all B' the algorithm returns "no". Figure 4 describes this algorithm more detailed.

Algorithm : FPT-$\text{IDPP}_{\text{leafs}\leq 2}$
Input : An incomplete haplotype matrix B.
Output : "yes" if $B \in \text{IDPP}_{\text{leafs}\leq 2}$, "no" otherwise.

```
1  delete rows and columns without 1-entries
2  choose row r and column c with r[c] = 1
3  for all completions of r and c do
4      B' ← corresponding partial completion of B
5      B'_0 ← induced submatrix of the rows R_0 and columns C_0 that are connected to r in G^1_{B'}
6      B'_1 ← induced submatrix of the rows R_1 and columns C_1 that are not connected to r in G^1_{B'}
7      if B'_0 ∈ IDPP_sorted and B'_1 ∈ IDPP_sorted then return "yes"
8  return "no"
```

Figure 4: FPT-IDPPP solves $\text{IDPP}_{\text{leafs}\leq 2}$ in time $O(2^k nm^2)$ for $n \times m$ incomplete haplotype matrices where the number of missing entries per row plus the maximal number of missing entries per column is at most k

Theorem 3.16. $\text{IDPP}_{\text{leafs}\leq 2}$ *can be solved in time* $O(2^k nm^2)$ *where k is the number of missing entries per row plus the maximal number of missing entries per column.*

Proof. First we show that FPT-$\text{IDPP}_{\text{leafs}\leq 2}$ decides $\text{IDPP}_{\text{leafs}\leq 2}$. Then we deduce its runtime.

We claim that an incomplete haplotype matrix B lies in $\text{IDPP}_{\text{leafs}\leq 2}$ if, and only if, FPT-$\text{IDPP}_{\text{leafs}\leq 2}$ returns "yes". We consider only matrices where each row and each column contains a 1-entry since all-0-rows and all-0-columns can be append to a matrix without changing directed perfect phylogenies.

We start with the "if"-part of the proof. Assume that FPT-$\text{IDPP}_{\text{leafs}\leq 2}$ says "yes". That means there exists a partial completion B' for B and partitions of rows (R_0, R_1) and columns (C_0, C_1) such that the induced submatrices of R_0 and C_0 and of R_1 and C_1 lie in $\text{IDPP}_{\text{sorted}}$. Furthermore, the submatrices induced by R_0 and C_1 and by R_1 and C_0 contain no 1-entries, since, otherwise, the algorithm would have chosen different partitions. This witnesses $B \in \text{IDPP}_{\text{leafs}\leq 2}$ by Lemma 3.15.

For the "only-if"-part we assume $B \in \text{IDPP}_{\text{leafs}\leq 2}$, thus there exists a completion B'' of B that admits a directed perfect path phylogeny T. As in the proof of Lemma 3.15, each branch of the path phylogeny is a phylogenetic sorting for the submatrix that is induced by its rows and columns. The entries of B'' that do not belong to one of these submatrices are 0-entries. Thus, it suffice to show that the algorithm considers at least once the partitions of rows and columns that is given by the branches of T. Therefore let r' and c' be the completions of r and c in B'' and let B' be the partial completion of B with respect to r' and c'. We show that the algorithm chooses exactly the partitions that are induced by T whenever it considers B'. As the algorithm does, we look at the partitions relatively to the row r. If a column c^* is not connected with r in $G_{B'}$, the same holds for c. Together with the fact that c^* contains a 1-entry, there exist a row r^* with $r^*[c^*] = 1$, but $r[c^*] = 0$ and $r^*[c] = 0$. Since $r[c] = 1$, the columns c and c^* contain the submatrix $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$ in B' and must, therefore, lie in different branches. In a similar way we obtain that a row lies in a different branch than r whenever they are not connected. If a column or a row is connected with r, by Lemma 3.14, it lies in the same branch with r.

The asymptotic time complexity of FPT-$\text{IDPP}_{\text{leafs}\leq 2}$ is dominated by the iteration over completions of r and c. There are 2^k iterations where k is the maximal number of missing entries per row plus the maximal number of missing entries per column. The partition of rows and columns can be computed by using depth-first-search from the vertex r in $G^1_{B'}$. This needs at most $O(nm)$ time. For the if statement we

need to solve the IDPP$_{\text{sorted}}$ problem. By using the characterization from the proof of Theorem 2.1, this can be done in time $O(nm^2)$ since the graph $G_{B'}$, which is used for this task, contains m vertices and at most nm directed edges. Overall, we obtain that the algorithm solves IDPP$_{\text{leafs}\leq 2}$ in time $O(2^k nm^2)$. □

4 Conclusion

The results of the present paper show that studying the influence of tree topologies on the hardness of the IPPH problem is worthwhile. On the one hand, restricting the tree topology can make problems harder as is the case for IDPP whose complexity jumps from polynomial time to NP-complete. On the other hand, tree topologies provide the first parameter for which a theoretical analysis of an algorithm is possible that works on arbitrary instances of the IPPH problem.

The concept of a *weight order* has turned out to be very useful for the study of undirected perfect phylogenies. We suggest trying to apply this notions to other problem versions as well.

There are some open problems that we would like to suggest for further research. First, is IPPH$_{\text{leafs}\leq l}$ fixed-parameter tractable with respect to the maximal number of missing entries per rows? We know that this is the case for IDPP$_{\text{leafs}\leq 2}$. Second, we showed that IDPP is NL-hard and it is known that IDPP $\in$ P. Can we characterize the complexity of IDPP more precisely? Finally, the main open problem is to improve the fixed-parameter results for IPPH. In particular, is IPPH fixed-parameter tractable with respect to the maximal number of ?-entries per column?

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] C. J. Benham, S. Kannan, M. Paterson, and T. Warnow. Hen's teeth and whale's feet: Generalized characters and their compatibility. *Journal of Computational Biology*, 2(4):515–525, 1995.

[3] A. G. Clark. Inference of haplotypes from PCR-amplified samples of diploid populations. *Journal of Molecular Biology and Evolution*, 7(2):111–22, 1990.

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

[5] Z. Ding, V. Filkov, and D. Gusfield. A linear-time algorithm for the perfect phylogeny haplotyping (PPH) problem. *Journal of Computational Biology*, 13(2):522–553, 2006.

[6] M. Elberfeld and T. Tantau. Computational complexity of perfect-phylogeny-related haplotyping problems. In *Proceedings of MFCS 2008*, volume 5162 of *Lecture Notes in Computer Science*, pages 299–310. Springer, 2008.

[7] 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.

[8] L. Excoffier and M. Slatkin. Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Molecular Biology and Evolution*, 12(5):921–7, 1995.

[9] D. Fallin and N. Schork. Accuracy of haplotype frequency estimation for biallelic loci via the expectation-maximation algorithm for unphased diploid genotype data. *American Journal of Human Genetics*, 67:947–959, 2000.

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

[11] J. Gramm, T. Hartman, T. Nierhoff, R. Sharan, and T. Tantau. On the complexity of snp block partitioning under the perfect phylogeny model. *Discrete Mathematics*, 2008. To appear, doi:10.1016/j.disc.2008.04.002.

[12] J. Gramm, A. Nickelsen, and T. Tantau. Fixed-parameter algorithms in phylogenetics. *The Computer Journal*, 2007.

[13] J. Gramm, T. Nierhoff, R. Sharan, and T. Tantau. Haplotyping with missing data via perfect path phylogenies. *Discrete and Applied Mathematics*, 155(6–7):788–805, 2007.

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

[15] D. Gusfield. Inference of haplotypes from samples of diploid populations: complexity and algorithms. *Journal of Computational Biology*, 8(3):305–23, 2001.

[16] D. Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. In *Proceedings of the Sixth Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 166–175. ACM Press, 2002.

[17] E. Halperin and R. M. Karp. Perfect phylogeny and haplotype assignment. In *Proceedings of the Sixth Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 10–19. ACM Press, 2004.

[18] M. Hawley and K. Kidd. Haplo: A program using the EM algorithm to estimate the frequency of multi-site haplotypes. *Journal of Heredity*, 86:409–41, 1995.

[19] L. Helmuth. Map of the human genome 3.0. *Science*, 293(5530):582–585, 2001.

[20] G. Kimmel and R. Shamir. The incomplete perfect phylogeny haplotype problem. *Journal of Bioinformatics and Computational Biology*, 3(2):359–384, 2005.

[21] Y. Liu and C.-Q. Zhang. A linear solution for haplotype perfect phylogeny problem. In *Proceedings of the International Conference on Advances in Bioinformatics and its Applications*, pages 173–184. World Scientific, 2005.

[22] I. Pe'er, T. Pupko, R. Shamir, and R. Sharan. Incomplete directed perfect phylogeny. *SIAM Journal on Computing*, 33(3):590–607, 2004.

[23] R. Vijaya Satya and A. Mukherjee. An optimal algorithm for perfect phylogeny haplotyping. *Journal of Computational Biology*, 13(4):897–928, 2006.

[24] R. Vijaya Satya and A. Mukherjee. The undirected incomplete perfect phylogeny problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2008. To appear, doi:10.1109/TCBB.2007.70218.

[25] M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9(1):91–116, 1992.