Haplotype Inference Constrained by Plausible Haplotype Data

Michael R. Fellows[1], Tzvika Hartman*[2], Danny Hermelin*[3], Gad M. Landau[4,5], Frances Rosamond[1], and Liat Rozenberg[4]

[1] The University of Newcastle, Newcastle - Australia
{mike.fellows,frances.rosamond}@cs.newcastle.edu.au
[2] Google, Tel-Aviv - Israel
tzvika@google.com
[3] Max Planck Institute for Informatics, Saarbrücken - Germany
hermelin@mpi-inf.mpg.de
[4] University of Haifa, Haifa - Israel.
liat@cri.haifa.ac.il, landau@cs.haifa.ac.il
[5] Polytechnic University, New York - USA.

Abstract. The *haplotype inference problem* (HIP) asks to find a set of haplotypes which resolve a given set of genotypes. This problem is important in practical fields such as the investigation of diseases or other types of genetic mutations. In order to find the haplotypes which are as close as possible to the real set of haplotypes that comprise the genotypes, two models have been suggested which are by now well-studied: The *perfect phylogeny* model and the *pure parsimony* model. All known algorithms up till now for haplotype inference may find haplotypes that are not necessarily plausible, *i.e.* very rare haplotypes or haplotypes that were never observed in the population. In order to overcome this disadvantage we study in this paper a new constrained version of HIP under the above mentioned models. In this new version, a pool of plausible haplotypes $\widetilde{H}$ is given together with the set of genotypes G, and the goal is to find a subset $H \subseteq \widetilde{H}$ that resolves G. For *constrained perfect phylogeny haplotyping* (CPPH), we provide initial insights and polynomial-time algorithms for some restricted cases of the problem. For *constrained parsimony haplotyping* (CPH), we show that the problem is fixed parameter tractable when parameterized by the size of the solution set of haplotypes.

Keywords: haplotyping, perfect phylogeny, pure parsimony, polynomial-time algorithms, parameterized complexity.

1 Introduction

Genetic information in living organisms is encoded in DNA sequences that are organized into *chromosomes*. Diploid organisms such as humans have two copies of every chromosome, which are not necessarily identical, with each copy called a *haplotype*. Identifying the common genetic variations that occur in humans is valuable in understanding diseases [1]. The genetic sequences of the population are almost totally identical, except from some bases that differ from one person to another with a frequency of more than some threshold (1% for example). Those differences are the common genetic variations and they are known as *single nucleotide polymorphisms* (SNPs).

The data described in each haplotype may be the full DNA, but it is more common to consider only the data of the SNPs, since the other sites are assumed to be identical. A *genotype* is the description of the two copies (haplotypes) together. When the two haplotypes agree, the site in the genotype has the agreed base. Such a site is called a *homozygous* site. When the two haplotypes disagree, the genotype has both bases, yet it does not tell which base occurs in which haplotype. This type of site is called *heterozygous*.

Current biological technologies give us an easier and cheaper way to obtain genotype data in comparison to haplotype data. However, the haplotype information is the one of greater use [19]. For this reason, it is necessary to computationally infer the

* Work done while at CRI, Haifa University, and Department of Computer Science, Bar-Ilan University.

haplotype information from the genotype data. An important biological fact is that almost always there are only two bases at an SNP, which can be marked as 0 and 1. A genotype will have 0 or 1 if the two haplotypes both have 0 or 1 in the same site respectively, or 2 otherwise (see Figure 1).

In view of that, a set of genotypes and a set of haplotypes can be represented as matrices. A *genotype matrix* is a matrix over $\{0, 1, 2\}$ where each row is a genotype and each column represents an SNP, and a *haplotype matrix* is a matrix over $\{0, 1\}$, where each row is a haplotype and each column represents an SNP.

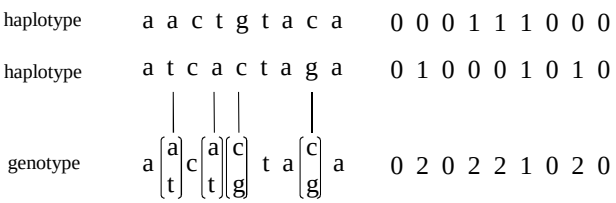

Fig. 1. Example of two haplotypes and the corresponding genotype. On the right are the same haplotypes and genotype using the 0,1,2 representation. Note that the figure simplifies matters a bit, since in general each column cannot be relabeled in the same way.

For the rest of the paper, let $g(i)$ represent the data at site i of genotype g, and $h(i)$ the data at site i of haplotype h. Furthermore, we will write $g(ij)$ (resp. $h(ij)$) instead of $g(i)g(j)$ (resp. $h(i)h(j)$).

Definition 1 (Resolution). *A pair of haplotypes $\{h, h'\}$ is said to* resolve *g if for each i: $g(i) = h(i)$ when $h(i) = h'(i)$, and $g(i) = 2$ otherwise. We extend this and say that a set of haplotypes H* resolves *a set of genotypes G, if for each $g \in G$, there is a pair $\{h, h'\} \in H$ which resolves g. The pair $\{h, h'\}$ is called a* resolution *of g, and H is a* resolution *of G.*

Definition 2 (Haplotype Inference Problem (HIP)). *Given a set of ℓ genotypes G, each of length m, find a resolution of G.*

Note that a genotype having $d \leq m$ heterozygous sites (sites marked with 2) has 2^{d-1} possible resolving pairs of haplotypes. The goal is to find the set of pairs which are as close as possible to the real set of haplotypes that created the genotype. Currently, there are two models used in practice that give two different biologically motivated heuristics on how to determine this:

1. **Perfect Phylogeny:** The perfect phylogeny model is a coalescent model which assumes no recombination. This means that the history of the haplotypes is represented as a tree where two haplotypes from two individuals have at most one recent common ancestor [19] (see [17, 19, 24, 32] for further information). Formally, a set of haplotypes (binary sequences) of length m *defines a perfect phylogeny* if the haplotypes appear as labels of a rooted tree which obeys the following properties [12]:
   - Each vertex of the tree is labeled by a binary sequence of length m representing a possible haplotype;
   - Every edge (u, v) is marked with i, where the base at site i in sequence u is different from the one in sequence v. Every coordinate i labels at most one edge.

   A common way of checking whether a set of haplotypes defines a perfect phylogeny is to check whether it obeys the *four gamete test*, *i.e.* the corresponding haplotype matrix does not contain, in any two columns, the *forbidden gamete submatrix*

   $$\begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 1 & 1 \end{pmatrix}.$$

   See Figure 2 for an example of a perfect phylogenetic tree, and the corresponding haplotype matrix. The *Perfect Phylogeny Haplotyping* (PPH) problem is the problem of finding for a given set of genotypes a resolution which defines a perfect phylogeny, whenever such a resolution exists.

2. **Pure Parsimony:** The pure parsimony model seeks the minimum set of haplotypes that resolves a given set of genotypes. The biological motivation behind this is the statistical observation that the number of distinct haplotypes in the population is vastly small [18, 19]. The *Parsimony Haplotyping* (PH) problem is the problem of finding a resolution of smallest size possible for a given set of genotypes.

In [17], Gusfield showed that the PPH problem is solvable in $O(nm\alpha(nm))$ time, where α is the inverse Ackerman function. Gusfield also showed a linear-time algorithm to build, once the first solution is found, a linear-space data structure that represents

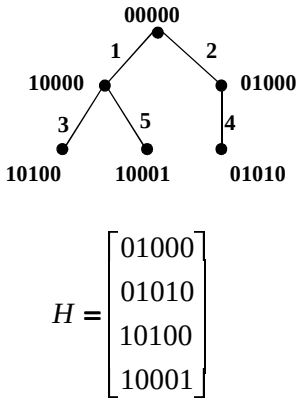

Fig. 2. Example of a perfect phylogenetic tree for the haplotypes $h_1 = (01000), h_2 = (01010), h_3 = (10100), h_4 = (10001)$. The matrix H is the haplotype matrix of these four haplotypes, and it obeys the four gamete test.

all PPH solutions. However, his work is based on complex graph-theoretic algorithms which are difficult to implement [19]. In [4, 12], algorithms fine-tuned to the actual combinatorial structure of the PPH problem were shown. These algorithms run in $O(nm^2)$ time and are easy to understand and implement. They also give a representation of all PPH solutions. More recent work developed $O(nm)$ time algorithms: In [9], the algorithm is graph-theoretic and uses a directed rooted graph called a "shadow tree", and in [30], the algorithm is based on inter-dependencies among the pairs of SNPs, and builds a data structure called "FlexTree" to represent all PPH solutions. Other works have researched different variations of PPH [5, 8, 14, 23].

The parsimony haplotyping (PH) problem was first suggested and proved to be NP-hard by Earl Hubell (unpublished). Gusfield formally introduced the problem in [18], and proposed an integer linear programming solution. More integer linear programming solutions following this were proposed in [6, 18, 21, 27]. Approximation algorithms for the problem were presented in [27, 28], and in [33], a branch-and-bound algorithm was proposed. Other theoretical results were shown in [25, 27, 31]. Most notably is the work of Sharan *et. al* [31], who characterized restricted instances of PH under the term $(\alpha, \beta) - bounded$, where α and β stand for the maximum number of heterozygous sites per row and column of the genotype matrix. Sharan *et al.* also showed that the PH problem is *fixed parameter tractable* (see [10] for a formal definition) when parameterized by the number of k haplotypes in the resolution of G. Many other works have researched different variations of the haplotyping problem [15, 16, 20, 22, 26, 29].

Iersel *et al.* continued in [25] to explore the bounded instances of the PH problem and of another related problem *Minimum Perfect Phylogeny Haplotyping* (MPPH), which looks for the minimum number of haplotypes that resolve a given set of genotypes and also define a perfect phylogeny. The MPPH problem was proved to be NP-hard [3]. The known results up till now are: For the PH problem, the (3,*) instance is APX-hard [27], the (4,3) instance is APX-hard [31], the (3,3) instance is APX-hard [25], the (2,*) instance is polynomial-time solvable [7, 28], and the (*,1) instance is also polynomial-time solvable [25]. For the MPPH problem, the (3,3) instance is APX-hard, and the (2,*) and (*,1) instances are polynomial-time solvable [25]. In both problems, the (*,2) instance was left as open problem, but it was shown that it polynomial-time solvable in a special structure of the genotype matrix [25, 31].

All known algorithms up till now for haplotype inference find resolutions for a given set of genotypes from the superset of all possible haplotypes (*i.e.* all m-length binary vectors). However, these algorithms may find resolutions that include binary vectors representing haplotypes that do not actually occur in the population, or are otherwise very rare. It is therefore biologically interesting to force the resolving haplotypes to be chosen only from a specific pool that contains only *plausible* haplotypes, *i.e.* haplotypes which have already been observed in relatively high frequencies in previous experiments. This pool can be determined by empirically setting up some statistical threshold, or by any other reasonable method.

In view of all this, we study in this paper a new constrained variant of the haplotype inference problem, in which a pool of plausible haplotypes $\widetilde{H}$ is given alongside the set of genotypes G, and the goal is to find a resolution of G which is a subset of $\widetilde{H}$.

Definition 3 (Constrained Haplotype Inference Problem (CHIP)). *Given a set of ℓ distinct genotypes G, each of length m, and a pool of n distinct plausible haplotypes $\widetilde{H}$ for G, each of length m, find a resolution $H \subseteq \widetilde{H}$ of G.*

The constrained perfect phylogeny haplotyping (CPPH) problem and the constrained parsimony haplotyping (CPH) problem are defined accordingly. Note that if $\ell > n(n-1)/2$ in the above definition, there is no solution possible, since by taking the entire pool of n plausible haplotypes we can resolve at

most $n(n-1)/2$ genotypes. On the other hand, there is no inequality necessarily in the other direction. We therefore assume $\ell \leq n(n-1)/2$ throughout the paper.

Note that CPH can be easily shown to be NP-hard by the straightforward reduction from $(3,*)$-bounded PH (shown to be NP-hard in [27]) which designates $\widetilde{H}$ to be all possible haplotypes resolving genotypes in G. At the time that this paper was in review, we did not know the complexity of CPPH. Recently, Elberfeld and Tantau should this problem is polynomial [11]. Here we present polynomial-time algorithms with slightly slower running-times for the (α,β) bounded cases of (*,1), (2,*), (5,2), and (3,3). In the second part of the paper we show that like PH [31], CPH is fixed-parameter tractable when parameterized by the number of haplotypes in a minimum size resolution $H \subseteq \widetilde{H}$ of G. The running-time of this algorithm was improved recently by Fleischer *et al.* [13].

2 Polynomial-Time Special Cases of CPPH

In this section we describe polynomial-time algorithms for CPPH with genotype matrices of specific structures. In [31], bounded cases of genotype matrices were introduced in order to explore the complexity of PH. The bounded cases were defined as follows:

Definition 4 ((α,β)-bounded [31]). *A genotype matrix G is* (α,β)-bounded *if it has at most α 2's per row and at most β 2's per column. Either α or β might be * which means there is no bound on the number of 2's per row or column, respectively.*

Here we use the same term of (α,β)-bounded to present polynomial-time algorithms for special cases of CPPH. We will present algorithms for the following cases: $(*,1)$, $(2,*)$, $(5,2)$, and $(3,3)$.

We begin with the following lemma which lists six matrices that we can assume G does not contain as submatrices, since including any one of these implies that all resolutions of G necessarily include the forbidden gamete submatrix. Note that we allow permutating rows and columns in each of the submatrices. The proof of the lemma is left to the reader.

Lemma 1. *If G includes one of the following 2×3 submatrices:*

$$\begin{pmatrix}2&0\\0&2\\1&1\end{pmatrix}, \begin{pmatrix}2&1\\1&2\\0&0\end{pmatrix}, \begin{pmatrix}2&0\\1&2\\0&1\end{pmatrix}, \begin{pmatrix}2&1\\0&0\\1&0\end{pmatrix}, \text{ or } \begin{pmatrix}2&0\\1&1\\0&1\end{pmatrix},$$

or the following 2×2 submatrix $\left(\begin{smallmatrix}2&0\\2&1\end{smallmatrix}\right)$, then G does not have a perfect phylogenic resolution.

As in Eskin *et al.* [12], we will be working with pairs of columns in G. Pairs of sites of a genotype can be split into two types, according to the data in these sites. Type I includes the pairs of sites that have only one possible resolution. These pairs of sites are (00), (01), (11), (20), and (21). The resolutions of these sites are described in the following list:

1.$(00) \rightarrow \left(\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right)$ 2.$(01) \rightarrow \left(\begin{smallmatrix}0&1\\0&1\end{smallmatrix}\right)$ 3.$(11) \rightarrow \left(\begin{smallmatrix}1&1\\1&1\end{smallmatrix}\right)$
4.$(20) \rightarrow \left(\begin{smallmatrix}0&0\\1&0\end{smallmatrix}\right)$ 5.$(21) \rightarrow \left(\begin{smallmatrix}0&1\\1&1\end{smallmatrix}\right)$

Type II includes pairs of sites with (22) (*22-columns*). A pair of 22-columns has two potential resolutions: $(22) \rightarrow \left(\begin{smallmatrix}0&0\\1&1\end{smallmatrix}\right)$, which will be called an *equal resolution*, or $(22) \rightarrow \left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)$, which will be called an *unequal resolution*.

Determining whether there is a perfect phylogenic resolution of G boils down to deciding the resolution type, equal or unequal, for every pair of 22-columns. For some 22-columns, the type of the resolution is determined by the given set of genotypes, for others it is determined by the given set of haplotypes, and for the rest we need algorithms that will find the proper resolution.

2.1 Preprocessing

We next present a preprocessing stage which is performed before all algorithms regardless of the specific structure of the input sets of genotypes or haplotypes.

A 22-columns ij must be resolved equally if the given set of genotypes G includes at least one of the following submatrices in columns ij: $\left(\begin{smallmatrix}0&0\\1&1\end{smallmatrix}\right), \left(\begin{smallmatrix}2&0\\1&2\end{smallmatrix}\right), \left(\begin{smallmatrix}2&0\\1&1\end{smallmatrix}\right)$, or $\left(\begin{smallmatrix}2&1\\0&0\end{smallmatrix}\right)$, since any resolution of G must include the combinations "00" and "11" in this case. For a similar reason columns ij must be resolved unequally when G includes at least one of the submatrices $\left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right), \left(\begin{smallmatrix}2&0\\0&2\end{smallmatrix}\right), \left(\begin{smallmatrix}2&1\\1&2\end{smallmatrix}\right), \left(\begin{smallmatrix}2&1\\1&0\end{smallmatrix}\right)$, or $\left(\begin{smallmatrix}2&0\\0&1\end{smallmatrix}\right)$. We will call these type of constraints on the resolution type *genotype constraints*. In addition, a 22-columns ij must be resolved equally (unequally) if the haplotype set includes for some genotype only equal (unequal) resolutions. These type of constraints will be called *haplotype constraints*.

The preprocessing ensures that haplotype-pairs which violate the above constraints will not be chosen. For each genotype $g_i \in G$, we use $\widetilde{H}(g_i)$ to denote all possible resolutions of g_i in $\widetilde{H}$, *i.e.* $\widetilde{H}(g_i) = \{\{h,h'\} \mid h,h' \in \widetilde{H}, h \text{ and } h' \text{ resolve } g_i\}$. The preprocessing step includes the following four steps:

1. Check whether the genotype matrix G can be resolved in a perfect phylogenic way (use any algorithm from [4, 9, 12, 30]). If not, report there is no solution.
2. For each genotype $g_i \in G$, $1 \leq i \leq \ell$, go over all pairs of haplotypes from $\widetilde{H}$ and compute $\widetilde{H}(g_i)$.
3. For each genotype constraint, delete from the sets $\widetilde{H}(g_1), \ldots, \widetilde{H}(g_\ell)$ all haplotype pairs that violate the constraint, *i.e.* all haplotype pairs that resolve the relevant sites in a different way than the constraint indicates.
4. For each haplotype constraint, delete from the sets $\widetilde{H}(g_1), \ldots, \widetilde{H}(g_\ell)$ all haplotype pairs that violate it. Note that the deletion of haplotypes may create new haplotype constraints. Repeat Step 4 until there is no change in the haplotype constraints.

After each step of steps 2 to 4 in the preprocessing stage, if one of the haplotypes sets $\widetilde{H}(g_1), \ldots, \widetilde{H}(g_\ell)$ becomes empty, it means there is no solution, and we are done. Once the preprocessing is complete, our goal is to find a resolution $H \subseteq \widetilde{H}$ of G, by selecting one pair of haplotypes from each $\widetilde{H}(g)$, $g \in G$. From here on out, we will only be concerned with resolutions of this type. In other words, we will use $\widetilde{H}$ as the set $\bigcup_{g \in G} \widetilde{H}(g)$ for the remainder of the section. Note that even after the preprocessing stage, not all resolutions in $\widetilde{H}$ will define a perfect phylogeny. This is because we are still left with 22-columns that have yet been resolved, as there might be two different genotypes g and g' which share a common pair of 22-columns ij, and $\widetilde{H}(g)$ and $\widetilde{H}(g')$ includes both resolutions for ij (*i.e.* equally and unequally). Such a pair of resolutions is said to be *conflicting*, and more generally, any pair of resolutions $\{h_1, h_1'\}$ and $\{h_2, h_2'\}$ are *conflicting* if $\{h_1, h_1', h_2, h_2'\}$ does not define a perfect phylogeny. We have the following two important lemmas:

Lemma 2. *Let $\{h_1, h_1'\}, \ldots, \{h_r, h_r'\}$ be resolutions of r (not necessarily distinct) genotypes in G after the preprocessing stage. Then $\{h_1, h_1'\}, \ldots, \{h_r, h_r'\}$ are pairwise non-conflicting iff $H = \bigcup_{1 \leq i \leq r} \{h_i, h_i'\}$ defines a perfect phylogeny.*

Proof. Let $G' \subseteq G$ denote the subset of r genotypes which H resolves. If H defines a perfect phylogeny, then clearly $\{h_1, h_1'\}, \ldots, \{h_r, h_r'\}$ are pairwise non-conflicting. To prove the other direction of the lemma, it suffices to show that H does not contain the forbidden gamete matrix. Suppose by way of contraction that this is not the case. Then the forbidden gamete submatrix occurs in H at some pair of sites $i, j \in \{1, \ldots, m\}$. We consider three possible cases:

- There is no 2 occurring in column i nor column j in G'. But this means that G' cannot be resolved in a perfect phylogenic way, and so the preprocessing algorithm would have halted at step 1 and reported that no solution can be found.
- G' has a 2 occurring at one of the two columns i or j, say column i, but there is no pair of 22-columns occurring at ij. Let $g \in G'$ be a genotype with $g(i) = 2$. Then either $g(j) = 1$ or $g(j) = 0$, so suppose first that $g(j) = 1$. We can assume w.l.o.g. that the forbidden gamete submatrix occurs in rows $\{h_a, h_b, h_c, h_d\}$, where $\{h_a, h_b\}$ is a resolution in $\{h_1, h_1'\}, \ldots, \{h_r, h_r'\}$ that resolves g. Furthermore, we know that $h_c(ij) = 10$ and $h_d(ij) = 00$, and moreover, h_c and h_d belong to two different resolutions in $\{h_1, h_1'\}, \ldots, \{h_r, h_r'\}$, as otherwise $\{h_a, h_b\}$ and $\{h_c, h_d\}$ would be conflicting. Thus, let $g', g'' \in G'$ be the two distinct genotypes resolved by the resolutions in $\{h_1, h_1'\}, \ldots, \{h_r, h_r'\}$ involving h_c and h_d. We know that $g'(ij) \in \{10, 20, 02\}$ and $g''(ij) \in \{00, 20, 02\}$. From this it is not difficult to verify that G' either contains the submatrix $\left(\begin{smallmatrix} 2 & 1 \\ 2 & 0 \end{smallmatrix}\right)$ at rows gg', gg'', or $g'g''$, and columns ij, or it contains one of the following three submatrices at rows $gg'g''$ and columns ij:

  $$\left(\begin{smallmatrix} 2 & 1 \\ 1 & 2 \\ 0 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 2 & 1 \\ 0 & 2 \\ 1 & 0 \end{smallmatrix}\right), \text{ or } \left(\begin{smallmatrix} 2 & 1 \\ 0 & 0 \\ 1 & 0 \end{smallmatrix}\right).$$

  In either case, any one of these submatrix implies that G' has no solution (Lemma 1), and the preprocessing stage would have halted at its first step. The case where $g(j) = 0$ is similar.
- G' has a pair of 22-columns at ij occurring in some row $g \in G'$. Again, we can assume w.l.o.g. that the forbidden gamete submatrix occurs in rows $\{h_a, h_b, h_c, h_d\}$, where $\{h_a, h_b\}$ is a resolution in $\{h_1, h_1'\}, \ldots, \{h_r, h_r'\}$ that resolves g. Suppose first that h_a and h_b resolve ij equally, *i.e.* $h_a(ij) = 11$ and $h_b(ij) = 00$. Then $h_c(ij) = 01$ and $h_d(ij) = 10$, and we know that h_c and h_d both belong to two different resolutions in $\{h_1, h_1'\}, \ldots, \{h_r, h_r'\}$. Let $g', g'' \in G'$ be the two genotypes that h_c and h_d resolve. Then it must be that $g'(ij) \in \{01, 02, 21\}$ and $g''(ij) \in \{10, 20, 12\}$. From this it is not hard to verify that G' must include one of the following six submatrices at rows $g'g''$ and columns ij:

  $$\left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 0 & 2 \\ 2 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 2 & 1 \\ 1 & 2 \end{smallmatrix}\right), \left(\begin{smallmatrix} 0 & 1 \\ 1 & 2 \end{smallmatrix}\right), \left(\begin{smallmatrix} 0 & 1 \\ 2 & 0 \end{smallmatrix}\right), \text{ or } \left(\begin{smallmatrix} 0 & 2 \\ 1 & 2 \end{smallmatrix}\right).$$

If G' includes the last submatrix, then G' does not have a perfect phylogenic resolution in the first place (Lemma 1), and so the preprocessing stage would have reported "no solution". If G' includes one of the first five submatrices, then there is a genotype constraint on ij stating that it must be resolved unequally, and so $\{h_a, h_b\}$ would have been removed from $\widetilde{H}(g)$ at step 3 of the preprocessing stage. In both cases we reach a contradiction. The case where $\{h_a, h_b\}$ are resolved unequally is similar.

In all three cases we have reached a contradiction to the fact the preprocessing stage did not halt, and so the lemma is proven. □

Lemma 3. *The preprocessing stage takes $O(m^4n^2 + m^2n^4)$ time.*

Proof. The first step of the preprocessing takes $O(\ell m)$-time by using one of the algorithms in [9, 30], which is $O(mn^2)$ when recalling that $\ell < n^2$. The second step takes $O(mn^4)$ time, since checking whether a specific haplotype pair resolves a specific genotype takes $O(m)$ time, and there are $\ell = O(n^2)$ genotypes and $O(n^2)$ haplotype pairs. The third step takes $O(m^2n^2)$-time, since finding one genotype constraint and deleting all violating haplotype pairs takes $O(\ell + n^2) = O(n^2)$ time, and this is done for each column pair of which we have at most $O(m^2)$ of. The last step takes $O(m^2n^2 \cdot \min\{m^2, n^2\})$, since finding all haplotype constraints and deleting all violating haplotype pairs takes $O(m^2n^2)$ time, and we repeat this at most $\min\{m^2, n^2\}$ times. This is because each one of the $O(m^2)$ column column pairs may create a new constraint only once, and since we cannot delete more than n^2 haplotype pairs. Thus, the total running time of the preprocessing stage is $O(m^2n^2 \cdot \min\{m^2, n^2\} + mn^4) = O(m^4n^2 + m^2n^4)$. □

2.2 The dependency graph

The definition of conflicting resolutions brings us to the notion independency and dependency between genotypes. Loosely speaking, a dependency between two genotypes $g, g' \in G$ arises when the decision on how to resolve g affects the decision on how to resolve g'. This obviously happens when there is a resolution $\{h_1, h'_1\} \in \widetilde{H}(g)$ conflicting with a resolution $\{h_2, h'_2\} \in \widetilde{H}(g')$. In this case we say that g and g' are *directly dependent*. If there is no resolution in $\widetilde{H}(g)$ conflicting with a solution in $\widetilde{H}(g')$, we say that g and g' are *not directly dependent*.

We next introduce the *dependency graph* $DG(G)$ of our given set of genotypes G, after they have been preprocessed by the algorithm in the previous section. Later on, we will use the properties of the dependency graph in our polynomial algorithms.

Definition 5 (dependency graph). *The* dependency graph $DG(G)$ *of a set of genotypes G is a graph which has a vertex for each genotype $g \in G$, and edge between vertices representing directly dependent genotypes.*

Lemma 4. *After the preprocessing stage, two genotypes that do not have any pair of 22-columns in common are not directly dependent.*

Proof. Consider two genotypes $g, g' \in G$ that don't have any common pair of 22-columns. Suppose by way of contradiction that there is a resolution $\{h_x, h_y\} \in \widetilde{H}(g)$ conflicting with a $\{h_{x'}, h_{y'}\} \in \widetilde{H}(g')$. This means that there is a pair of columns ij of $\{h_x, h_y, h_{x'}, h_{y'}\}$ that has the forbidden gamete matrix. But this can only happen when G includes (in the rows gg', and in columns ij) the forbidden submatrix $\left(\begin{smallmatrix} 2 & 0 \\ 2 & 1 \end{smallmatrix}\right)$ of Lemma 1. □

Lemma 5. *Let G_1 and G_2 be two connected components in $DG(G)$ after the preprocessing stage has been performed. If $H_1 \subseteq \widetilde{H}$ and $H_2 \subseteq \widetilde{H}$ are perfect phylogenic resolutions of G_1 and G_2 respectively, then $H_1 \cup H_2$ is a perfect phylogenic resolution of $G_1 \cup G_2$.*

Proof. Consider any pair of resolutions in $H_1 \cup H_2$ of two genotypes $g, g' \in G_1 \cup G_2$. If g and g' are not both in G_1, nor in G_2, then there is no edge between them in $DG(G)$, meaning that they are not directly dependent. Hence, the pair of resolutions is non-conflicting by definition. If $g, g' \in G_1$ or $g, g' \in G_2$, then by Lemma 2, the pair of resolutions is non-conflicting as both H_1 and H_2 define a perfect phylogeny. It follows that all resolutions in $H_1 \cup H_2$ are pairwise non-conflicting, and so again by Lemma 2, $H_1 \cup H_2$ defines a perfect phylogeny. □

In view of Lemma 5, every connected component can be resolved individually, and the union of the chosen haplotypes will give the desirable resolution of G. As mentioned in Definition 4, a genotype matrix G is (α,β)-bounded when it has at most α 2s per row and at most β 2s per column, where α or β might be * to indicate there is no such limitation. The following algorithms refer to different values of α and

β, as the headlines indicate. Note that throughout the remaining subsections we assume that the preprocessing stage has been completed.

2.3 (*,1)- and (2,*)-bounded cases

Let us begin with the simple cases of (1,*)-bounded and (*,2)-bounded genotype matrices.

Lemma 6. *Any pair of distinct genotypes g_1 and g_2 in a $(*,1)$-bounded genotype matrix are not directly dependent.*

Proof. Since there is at most one 2 per column, g_1 and g_2 cannot have a 2 in the same column. Due to this, g_1 and g_2 can never share a pair of 22-columns, which according to Lemma 4, means they cannot be directly dependent. □

According to Lemma 6, if G is a $(*,1)$-bounded genotype matrix, then the dependency graph $DG(G)$ has no edges, and its connected components are of size one. The algorithm will choose for H one pair of haplotypes from every set $\widetilde{H}(g)$, $g \in G$, and H will define a perfect phylogeny according to Lemma 5.

Lemma 7. *Any pair of distinct genotypes g_1 and g_2 in a $(2,*)$-bounded genotype matrix are not directly dependent.*

Proof. According to Lemma 4, two genotypes are directly dependent only if they have a pair of 22-columns in common. If g_1 and g_2 are distinct genotypes in a $(2,*)$-bounded genotype matrix, and they have a pair of 22-columns in common, it means that G contains at the forbidden submatrix $\left(\begin{smallmatrix}2&0\\2&1\end{smallmatrix}\right)$ of Lemma 1, and so at the first step of the preprocessing stage we would have determined that there is no solution. □

According to Lemma 7, if G is a $(2,*)$-bounded genotype matrix, then the dependency graph $DG(G)$ has no edges, and its connected components are of size one. The algorithm for solving the $(2,*)$-bounded case is thus identical to the above algorithm for the $(*,1)$-bounded case.

Theorem 1. CPPH *with $(*,1)$- or $(2,*)$-bounded genotype matrices is solvable in $O(m^4n^2 + m^2n^4)$ time.*

Proof. According to the above, in the case of $(*,1)$- or $(2,*)$-bounded genotype matrices G, one can determine a perfect phylogenetic resolution for G by picking a resolution pair from each $\widetilde{H}(g)$, $g \in G$. Since this can be done in $O(\ell)$ time, the preprocessing stage dominates the running time of the entire algorithm, which according to Lemma 3 is $O(m^4n^2+m^2n^4)$. □

2.4 (5,2)-bounded case

It is convenient to mark every edge gg' in the dependency graph $DG(G)$ with the indices of the 2-columns g and g' share. Observe that in the $(*,2)$-bounded case, a specific index will appear only on one edge of the dependency graph, since there are no more than two genotypes that have 2 in the same column. Thus, in the $(\alpha,2)$-bounded case, each genotype $g \in G$ can be adjacent to at most $\lfloor \alpha/2 \rfloor$ other genotypes in $DG(G)$, since according to Lemma 4, each adjacency means that g shares a pair of 22-columns with his neighbor genotype. We thus have the following important property in the $(5,2)$-bounded case:

Lemma 8. *If G is a (5,2)-bounded genotype matrix then $DG(G)$ has maximum degree 2.*

The lemma above implies that if G is a $(5,2)$-bounded genotype matrix, then every connected component in $DG(G)$ is either a path or a cycle. In the following lemma, we show that resolving a path or a cycle in $DG(G)$ reduces to solving a 2-SAT instance for which we have a well-known polynomial (even linear) time algorithm.

Lemma 9. *If G' is a connected component of $DG(G)$ which is either a path or cycle, then one can determine whether G' has perfect phylogenetic resolution $H' \subseteq \widetilde{H}$ in $O(sn^2)$ time, where s is the sum over all s_e number of column labels on edges e of $DG(G')$.*

Proof. Let G' be connected component in $DG(G)$ which is either a path or a cycle. As a first step, we get rid of all edges that have more than two labels by subdividing these edges. That is, if gg' is an edge in G' labeled by $l_1, \ldots, l_t$, $t > 2$, then we replace this edge by the path $g_1, g_2, \ldots, g_{\lceil t/2 \rceil}, g'$, where g_i, g_{i+1} is labeled with l_{2i-1}, l_{2i} for $i \in \{1, \ldots, \lceil t/2 \rceil - 1\}$, and $g_{\lceil t/2 \rceil}, g'$ is labeled with l_{t-1}, l_t. (Observe that in the $(5,2)$-bounded case $t \leq 4$, and in fact $t \leq 3$ if we assume that G' has more than two genotypes.) We therefore henceforth assume that any edge in G' has two labels corresponding to a pair of 22-columns.

Let $g_1, \ldots, g_r$ denote the genotypes in G' according to their order of appearance in G' (*i.e.* g_i is adjacent to g_{i+1} for each $i \in \{1, \ldots, r-1\}$, and g_r is adjacent to g_1 if G' is a cycle). We construct a 2-CNF

boolean formula $\phi(G')$ corresponding to G', such that $\phi(G')$ is satisfiable iff there is a perfect phylogenic resolution for G' in $\widetilde{H}$. For each pair of 22-columns ij appearing on an edge in G', we assign a boolean variable x_{ij}, where we interpret $x_{ij} = 1$ (resp. $x_{ij} = 0$) to mean that ij should be resolved equally (resp. unequally). The clauses of $\phi(G')$ are constructed as follows: First, if G' is a path, and ab and cd are the labels on the edges g_1g_2 and $g_{r-1}g_r$ respectively, then we add the clause (x_{ab}) (resp. $(\neg x_{ab})$) to $\phi(G')$ if ab can only be resolved equally (resp. unequally) in $\widetilde{H}(g_1)$, and similarly, we add (x_{cd}) (resp. $(\neg x_{cd})$) if cd can only be resolved equally (resp. unequally) in $\widetilde{H}(g_r)$. Next, for every genotype g_i of degree two in $DG(G')$, where ab and cd be the 22-columns labeling the edges $g_{i-1}g_i$ and g_ig_{i+1} (or g_ig_1 in case $i = r$ and G' is a cycle), we proceed according to the following:

1. If there is no equal resolution for ab in $\widetilde{H}(g_i)$, then we add the clause $(\neg x_{ab})$ to $\phi(G')$.
2. If there is no unequal resolution for ab in $\widetilde{H}(g_i)$, then we add the clause (x_{ab}) to $\phi(G')$.
3. If every equal resolution for ab in $\widetilde{H}(g_i)$ is an equal resolution for cd, then we add the clause $(x_{ab} \vee \neg x_{cd})$ to $\phi(G')$.
4. If every equal resolution for ab in $\widetilde{H}(g_i)$ is an unequal resolution for cd, then we add the clause $(x_{ab} \vee x_{cd})$ to $\phi(G')$.
5. If every unequal resolution for ab in $\widetilde{H}(g_i)$ is an equal resolution for cd, then we add the clause $(\neg x_{ab} \vee \neg x_{cd})$ to $\phi(G')$.
6. If every unequal resolution for ab in $\widetilde{H}(g_i)$ is an equal resolution for cd, then we add the clause $(\neg x_{ab} \vee x_{cd})$ to $\phi(G')$.

It is not difficult to see that $\phi(G')$ is satisfiable iff G' has a perfect phylogenetic resolution $H' \subseteq \widetilde{H}$. Furthermore, the construction of $\phi(G')$ requires $O(\ell s) = O(sn^2)$ time, since we need at most $O(\ell)$ time for each edge in G', and there are at most s such edges. Also observe that $\phi(G')$ has at most $O(s)$ variables and clauses. Thus, using the linear-time algorithm of Aspvall, Plass, and Tarjan for determining whether a 2-CNF is satisfiable [2], we obtain the required time bound in the lemma. □

Theorem 2. CPPH *restricted to* $(5, 2)$*-bounded genotype matrices is solvable in* $O(m^4n^2 + m^2n^4)$ *time.*

Proof. According to Lemma 5 and Lemma 8, solving CPPH in the $(5, 2)$-bounded case boils down to resolving connected components in $DG(G)$ which are either paths or cycles. According to Lemma 9, this can be done in $O(sn^2)$ time for any such component G', where s is the total number of column labels that appear on edges of G'. Since the total number of edge labels in $DG(G)$ is at most m in the $(5, 2)$-bounded case, the total time required to resolve all connected components of $DG(G)$ is $O(mn^2)$. Accounting also for the time required by our preprocessing stage, we get the desired time bound of the theorem. □

2.5 (3,3)-bounded case

We next turn to deal with the $(3, 3)$-bounded case. Similar to the $(5, 2)$-bounded case of the previous section, here every connected component of $DG(G')$ will have a very simple structure. We have the following lemma:

Lemma 10. *If G is a $(3, 3)$-bounded genotype matrix, then every connected component in $DG(G)$ is either a path, a cycle, or is of size at most four.*

Proof. Let g be some genotype which is directly dependent with three other distinct genotypes g_1, g_2, and g_3, and let abc denote the three columns with $g(a) = g(b) = g(c) = 2$. Then the submatrix of G with rows $\{g_1, g_2, g_3\}$ and the columns abc must be of the following form:

$$\begin{pmatrix} \{0,1\} & 2 & 2 \\ 2 & \{0,1\} & 2 \\ 2 & 2 & \{0,1\} \end{pmatrix}$$

Here $\{0, 1\}$ specifies that any one of the two values $\{0, 1\}$ can appear in the submatrix. It is not difficult to see that since G has at most three 2's in each column and each row, and since adjacency in $DG(G')$ requires sharing at least a pair of 22-columns (Lemma 4), none of the four genotypes g, g_1, g_2, and g_3 can be adjacent to any other genotype in $DG(G)$. It follows that any connected component in $DG(G)$ with a vertex of degree at least three has size at most four. The lemma thus follows from the fact that every connected component of maximum degree two is either a path or a cycle. □

Theorem 3. CPPH *restricted to* $(3, 3)$*-bounded genotype matrices is solvable in* $O(m^4n^2 + m^2n^4)$ *time.*

Proof. First observe that all connected components of $DG(G)$ which are paths or cycles can be resolved in the time bound given in the theorem (including the time required for preprocessing) using a similar

analysis as the one used in Theorem 2. Next note that since G has at most three 2's in each row, the number of possible resolutions for each genotypes is $O(1)$, and given a genotype $g \in G$, all these resolutions can be found in $O(m\ell) = O(mn^2)$ time by simply going through all haplotypes in $\widetilde{H}$. It follows that one can determine whether a connected component of size at most four in $DG(G)$ can be resolved in a perfect phylogenetic way in the same time complexity, *i.e.* $O(mn^2)$ time. Since there are at most $O(\ell) = O(n^2)$ components, the time bound in the theorem above follows. □

3 FPT Algorithm for CPH

We next consider the CPH problem. In [31], Sharan *et al.* showed that PH is fixed parameter tractable when parameterized by the size k of the minimum-size resolution of G. Essentially, this means that there is algorithm for PH running in $f(k) \cdot N^c$, where $f()$ is an arbitrary (typically super-polynomial) computable function, N is the total input size, and c is a constant independent of k (see [10] for a formal definition of a parameterized algorithm). Observe that this is, in general, much faster than the brute-force $N^{k+O(1)}$ algorithm. Here, we show an analogous result for CPH. Our approach involves solving a dynamic program to determine whether there is any $H \subseteq \widetilde{H}$ of size $\kappa \leq k$ which resolves G. Throughout the section we use G_i, $1 \leq i \leq \ell$, to denote the subset of genotypes $\{g_1, \ldots, g_i\} \subseteq G$.

Probably the first dynamic-programming solution to come to mind for CPH, is to compute all possible resolutions $H' \subseteq \widetilde{H}$ of G_i from the resolutions of G_{i-1}. However, the number of k-subsets resolving G_i might be $\Omega(n^k)$, which is too much in terms of a parameterized algorithm. We therefore take an alternate route. Instead of computing the actual subsets which resolve G_i, we will compute abstract "blueprints" of these subsets. But before we define these blueprints, let us briefly change our terminology a bit. For presentation purposes, it will be convenient to consider resolutions as ordered sequences rather than sets, where we also allow some haplotypes to appear more than once in the sequences. Thus, from here on out, we will think of a resolution of size κ as a sequence of haplotypes $(h_1, \ldots, h_\kappa)$, with possibly $h_i = h_j$ for some $i \neq j \in \{1, \ldots, \kappa\}$. The blueprints of such resolutions, which we call κ-plans, are formally defined as follows:

Definition 6 (κ-plan). *Let κ be an integer in $\{1, \ldots, k\}$. A* κ-plan *is a string of length $i \leq \ell$ over the alphabet $\{\{x, y\} \mid 1 \leq x \leq y \leq \kappa\}$.*

Let $H' = (h_1, \ldots, h_\kappa)$ be a resolution of $G_i = \{g_1, \ldots, g_i\}$ of size κ. A κ-plan p is *associated with H'* if when $\{x, y\}$ is the j'th letter in p, $1 \leq x \leq y \leq \kappa$ and $1 \leq j \leq i$, then h_x and h_y resolve g_j. We will say that p is *valid* for G_i if there is a resolution of G_i associated with p. In this way, a valid κ-plan does not describe the actual resolution of G_i, but it does provide all relevant information concerning which genotypes are resolved using the same haplotypes.

Definition 7 ($\mathrm{DP}[\kappa, i], \mathcal{DP}[\kappa, i]$). *Let κ be an integer in $\{1, \ldots, k\}$, and i be an integer in $\{1, \ldots, \ell\}$. We denote by $\mathcal{DP}[\kappa, i]$ the set of all κ-plans of length i, and by $\mathrm{DP}[\kappa, i] \subseteq \mathcal{DP}[\kappa, i]$ the set of all valid κ-plans for $G_i = \{g_1, \ldots, g_i\}$.*

Lemma 11. *$|\mathrm{DP}[\kappa, i]| \leq |\mathcal{DP}[\kappa, i]| \leq k^{O(k^2)}$ for any $\kappa \leq k$ and $i \leq \ell$.*

Proof. To prove the lemma, recall that $\ell < k^2$. The number of distinct strings of length at most k^2 over an alphabet of size at most $\binom{\kappa}{2} < k^2$ is bounded by $k^{O(k^2)}$. □

Our algorithm proceeds by computing $\mathrm{DP}[\kappa, i]$ in increasing values of κ and i. The base-cases of this computation are

1. $\mathrm{DP}[\kappa, 1] = \mathcal{DP}[\kappa, 1]$ for all $1 \leq \kappa \leq k$, and
2. $\mathrm{DP}[1, i] = \mathrm{DP}[2, i] = \emptyset$ for all $2 \leq i \leq \ell$.

Clearly, G can be resolved using $\kappa \leq k$ haplotypes if and only if $G_\ell = G$ has at least one valid κ-plan. Hence, assuming we can correctly compute $\mathrm{DP}[\kappa, i]$ for all $1 \leq \kappa \leq k$ and $1 \leq i \leq \ell$, the correctness of our algorithm is immediate. What remains to be described is the dynamic-programming step for computing $\mathrm{DP}[\kappa, i]$.

For this, we will first need to introduce some additional terminology. Let p be some κ-plan which is valid for G_i, and let $h, h' \in \widetilde{H}$ be some pair of (not necessarily distinct) haplotypes. For a given $x, y \in \{1, \ldots, \kappa\}$, we say that the *assignment of $h_x = h$ and $h_y = h'$ is compatible with p* if there is a resolution $H' = (h_1, \ldots, h_\kappa)$ of G_i associated with p such that $h_x = h$ and $h_y = h'$. We will need the following lemma.

Lemma 12. *Let p be a valid κ-plan for G_i, for some $i \in \{1, \ldots, \ell\}$, and let $h, h' \in \widetilde{H}$. Then given any*

pair of integers $x, y \in \{1, \ldots, \kappa\}$, one can determine whether the assignment of $h_x = h$ and $h_y = h'$ is compatible with p in $O(k^3 m)$ time.

Proof. The algorithm is an iterative process that proceeds as follows. At each step, the algorithm has a set D of indices representing all haplotypes which have already been determined by the algorithm. Thus, at each iteration, we assume that the algorithm has a specific haplotype h_d for each $d \in D$. At the beginning of the algorithm $D = \{x, y\}$, where $h_x = h$ and $h_y = h'$.

Now, at the start of each iteration, the algorithm goes through all indices $d \in D$, and for each such index, the algorithm marks all positions j, $1 \leq j \leq i$, with a letter $\{d, d'\}$ occurring in p. For each such position j, the algorithm computes $h_{d'}$ from g_j and h_d. If $d' \notin D$, then the algorithm has discovered a new haplotype (possibly identical to a previously discovered haplotype), and so it adds d' to D and starts another iteration. If $d' \in D$, there are two possible outcomes: Either $h_{d'}$ is equal to the haplotype assigned to index d' by the algorithm in the past, or not. In the first case, the algorithm continues its computation at this iteration, and in the latter the algorithm determines incompatibility. The algorithm terminates when it has either declared incompatibility, or when there are no new indices added to D in a given iteration. In this case, the algorithm declares compatibility.

The correctness of this algorithm is immediate when it declares incompatibility. Observe that in case the algorithm determines compatibility, it has not necessarily determined all haplotypes in a resolution corresponding to p. Nevertheless, since p is known to be a valid a κ-plan for G_i, the correctness in this case follows as well. As for its time complexity, notice that the algorithm performs at most κ iterations, with each iteration requiring $O(\ell m)$ time. Thus, the total time complexity of this algorithm is $O(\kappa \ell m)$, which is $O(k^3 m)$ since $\ell < k^2$. □

We are now in position to present the dynamic-programming computation of the table DP. The dynamic-programming step for computing $\mathrm{DP}[\kappa, i]$ proceeds as follows:

1. $\mathrm{DP}[\kappa, i] \leftarrow \mathrm{DP}[\kappa - 1, i]$.
2. For each $p \in \mathrm{DP}[\kappa - 2, i - 1]$:
   - Concatenate $\{\kappa, \kappa - 1\}$ to the end of p, and add this new κ-plan to $\mathrm{DP}[\kappa, i]$.
3. For each $h, h' \in \widetilde{H}$ resolving g_i, for each $p \in \mathrm{DP}[\kappa - 1, i - 1]$, and for each $x \in \{1, \ldots, \kappa - 1\}$:
   - Set $y = x$, and check whether the assignment of $h_y = h_x = h$ is compatible with p. If so, concatenate $\{x, \kappa\}$ to the end of p, and add this new κ-plan to $\mathrm{DP}[\kappa, i]$.
4. For each $h, h' \in \widetilde{H}$ resolving g_i, for each $p \in \mathrm{DP}[\kappa, i - 1]$, and for each $x \neq y \in \{1, \ldots, \kappa\}$:
   - Check whether the assignment of $h_x = h$ and $h_y = h'$ is compatible with p. If so, concatenate $\{x, y\}$ to the end of p, and add this new κ-plan to $\mathrm{DP}[\kappa, i]$.

Correctness of the dynamic programming step is straightforward. Indeed, any valid κ-plan p_0 of G_i is either a κ'-plan of G_i for some $\kappa' < \kappa$ (Line 1), or it can be decomposed into either:

- A valid $(\kappa - 2)$-plan p of G_{i-1} concatenated to a new letter $\{\kappa, \kappa - 1\}$ (Line 2). In this case, in any resolution $(h_1, \ldots, h_\kappa)$ associated with p_0, we know that h_κ and $h_{\kappa-1}$ resolve g_i, and that $(h_1, \ldots, h_{\kappa-2})$ resolves G_{i-1}.
- A valid $(\kappa - 1)$-plan p of G_{i-1} concatenated to a new letter $\{\kappa, x\}$ for some $1 \leq x \leq \kappa - 1$ (Line 3). In this case, in any resolution $(h_1, \ldots, h_\kappa)$ associated with p_0, h_κ and h_x resolve g_i, and $(h_1, \ldots, h_{\kappa-1})$ resolves G_{i-1}.
- A valid κ-plan p of G_{i-1} concatenated to a letter $\{x, y\}$ for some $1 \leq x < y \leq \kappa$ (Line 4). In this case, in any resolution $(h_1, \ldots, h_\kappa)$ associated with p_0, h_x and h_y resolve g_i, and $(h_1, \ldots, h_\kappa)$ also resolves G_{i-1}.

Theorem 4. CPH *is solvable in $k^{O(k^2)} n^2 m$ time, where k is the size of the required resolution of G.*

Proof. The description of the algorithm, and its correctness, is discussed above. Let us next analyze its time complexity. First note that the number of different entries in the dynamic programming entries DP is $O(k\ell)$, which is $O(k^3)$ since $\ell < k^2$. To compute the entry $\mathrm{DP}[\kappa, i]$, for $\kappa \in \{2, \ldots, k\}$ and $i \in \{2, \ldots, \ell\}$, we go through all four steps of the dynamic program described above. It is not difficult to see that the time required in fourth step dominates the time required for all the rest of the steps, so we analyze only this step. In the fourth step, we go through all pairs of haplotypes $h, h' \in \widetilde{H}$ that resolve g_i, through all κ-plans in $\mathrm{DP}[\kappa, i - 1]$, and through all pairs of indices $x \neq y \in \{1, \ldots, \kappa\}$, and for each such set of objects we invoke the algorithm given in Lemma 12 which runs in $O(k^3 m)$ time. Since the total number of κ-plans in $\mathrm{DP}[\kappa, i - 1]$ is bounded by $k^{O(k^2)}$ according to Lemma 3, and since checking whether a

pair of haplotypes resolves g_i requires $O(m)$ time, the total time complexity of this step is bounded by $O(n^2m) + n^2k^{O(k^2)}k^2 \cdot k^3m = k^{O(k^2)}n^2m$. Thus, the total time complexity of computing the entire DP table is $O(k^3) \cdot k^{O(k^2)}n^2m = k^{O(k^2)}n^2m$. □

4 Conclusions and Open Problems

In this paper we studied a new variant of the haplotype inference problem which we termed *Constrained Haplotype Inference*, where the haplotypes to be inferred are constrained to be a subset of a given pool of plausible haplotypes. We focused on two established models for haplotype inference, the perfect phylogeny model, and the pure parsimony model, and gave both positive and negative results for constrained haplotype inference under these models. We believe that our results could prove to be helpful as subroutines to quickly solve optimally sub-instances where the number of heterozygous sites are known to be limited, and/or the number of inferred haplotypes is expected to be relatively small. Also, we hope our results will provide insights into the general haplotype inference problem, which has yet to be understood completely. Finally, as a possible future direction of research, it would be useful to extend the results here and in succeeding papers [11, 13] to hold for weighted variants of the problems discussed.

Acknowledgements

We would like to thank an anonymous referee for pointing out a bug in the proof of Lemma 2 which appeared in an early version of the paper.

References

1. The international hapmap project. *Nature*, 426:789–796, 2003.
2. B. Aspvall, M.F. Plass, and R.E. Tarjan. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.
3. V. Bafna, D. Gusfield, S. Hannenhalli, and S.Yooseph. A note on efficient computation of haplotypes via perfect phylogeny. *Journal of Computational Biology*, 11:858–866, 2004.
4. V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph. Haplotyping as perfect phylogeny: A direct approach. *Journal of Computational Biology*, 10:323–340, 2003.
5. T. Barzuza, J.S. Beckmann, R. Shamir, and I. Peer. Computational problems in perfect phylogeny haplotyping: XOR-genotypes and tag SNPs. In *15th Annual Symposium on Combinatorial Pattern Matching (CPM)*, pages 14–31, 2004.
6. D. Brown and I.M. Harrower. A new integer programming formulation for the pure parsimony problem in haplotype analysis. In *Proceedings of Workshop on Algorithms in Bioinformatics (WABI)*, pages 254–265, 2004.
7. R. Cilibrasi, L. van Iersel, S. Kelk, and J. Tromp. On the complexity of several haplotyping problems. In *Proceedings of Workshop on Algorithms in Bioinformatics (WABI)*, pages 128–139, 2005.
8. P. Damaschke. Fast perfect phylogeny haplotype inference. In *14th Symposium on Fundamentals of Computation Theory (FCT)*, pages 183–194, 2003.
9. Z. Ding, V. Filkov, and D. Gusfield. A linear-time algorithm for the perfect phylogeny haplotyping (pph) problem. *Journal of Computational Biology*, 13:522–553, 2006.
10. R. Downey and M. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.
11. M. Elberfeld and T. Tantau. Phylogeny- and parsimony-based haplotype inference with constraints. In *21th Annual Symposium on Combinatorial Pattern Matching (CPM)*, 2010. To appear.
12. E. Eskin, E. Halperin, and R. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. *Journal of Bioinformatics and Computational Biology*, 1:1–20, 2003.
13. R. Fleischer, J. Guo, R. Niedermeier, J. Uhlmann, Y. Wang, M. Weller, and X. Wu. Extended islands of tractability for parsimony haplotyping. In *21th Annual Symposium on Combinatorial Pattern Matching (CPM)*, 2010. To appear.
14. J. Gramm, T. Nierhoff, R. Sharan, and T. Tantau. On the complexity of haplotyping via perfect phylogeny. In *Proceedings of RECOMB Satellite Workshop on Computational Methods for SNPs and Haplotypes*, 2004.
15. J. Gramm, T. Nierhoff, R. Sharan, and T. Tantau. Haplotyping with missing data via perfect path phylogenies. *Discrete Applied Mathematics*, 155:788–805, 2007.
16. G. Greenspan and D. Geiger. Model-based inference of haplotype block variation. In *Research in Computational Molecular Biology (RECOMB '03)*, pages 131–137, 2003.
17. D. Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions (extended abstract). In *Proceedings of RECOMB*, pages 166–175, 2002.
18. D. Gusfield. Haplotype inference by pure parsimony. In *14th Annual Symposium on Combinatorial Pattern Matching (CPM)*, pages 144–155, 2003.

19. D. Gusfield and S.H. Orzack. Haplotype inference. *In CRC Handbook on Bioinformatics (Editor S. Aluru)*, 2005.
20. D. Gusfield, Y. Song, and Y. Wu. Algorithms for imperfect phylogeny haplotyping with a single homoplasy or recombination event. In *Proc Workshop on Algorithms in Bioinformatics (WABI)*, pages 152–164, 2005.
21. B. Halldórsson, V. Bafna, N. Edwards, R. Lippert, S. Yooseph, and S. Istrail. A survey of computational methods for determining haplotypes. In *Proceedings of RECOMB Satellite on Computational Methods for SNPs and Haplotype Inference*, pages 26–47, 2003.
22. E. Halperin and E. Eskin. Haplotype reconstruction from genotype data using imperfect phylogeny. *Bioinformatics*, 20:1842–1849, 2004.
23. E. Halperin and R.M. Karp. Perfect phylogeny and haplotype assignment. In *Proceedings of RECOMB*, pages 10–19, 2004.
24. R. Hudson. Gene genealogies and the coalescent process. *Oxford Survey of Evolutionary Biology*, 7:1–44, 1990.
25. L. Van Iersel, J. Keijsper, S. Kelk, and L. Stougie. Beaches of islands of tractability: Algorithms for parsimony and minimum perfect phylogeny haplotyping problems. In *Proceedings of Workshop on Algorithms in Bioinformatics (WABI)*, pages 80–91, 2006.
26. G. Kimmel and R. Shamir. The incomplete perfect phylogeny haplotype problem. *Journal of Bioinformatics and Comutational Biology*, 3:359–384, 2005.
27. G. Lancia, C. Pinotti, and R. Rizzi. Haplotyping population by pure parsimony: Complexity, exact and approximation algorithms. *INFORMS Journal on Computing, special issue on Comutational Biology*, 16:348–359, 2004.
28. G. Lancia and R. Rizzi. A polynomial case of the parsimony haplotyping problem. *Operations Research Letters*, 34:289–295, 2006.
29. P. Rastas, M. Koivisto, H. Mannila, and E. Ukkonnen. A hidden markov technique for haplotype reconstruction. In *Algorithms in Bioinformatics (WABI)*, pages 140–151, 2005.
30. R.V. Satya and A. Mukherjee. An optimal algorithm for perfect phylogeny haplotyping. *Journal of Computational Biology*, 13(4):897–928, 2006.
31. R. Sharan, B. Halldorsson, and S. Istrail. Islands of tractability for parsimony haplotyping. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3:303–311, 2006.
32. S. Tavare. Calibrating the clock: Using stochastic process to measure the rate of evolution. *In E. Lander and M. Waterman, editors. Calculating the Secrets of Life*, 1995.
33. L. Wang and L. Xu. Haplotype inference by maximum parsimony. *Bioinformatics*, 19:1773–1780, 2003.