

Extended Islands of Tractability for Parsimony Haplotyping*

Rudolf Fleischer¹, Jiong Guo², Rolf Niedermeier³, Johannes Uhlmann³, Yihui Wang¹,
Mathias Weller³, Xi Wu¹

¹ School of Computer Science, IIPL, Fudan University, Shanghai, China
[rudolf, yihuiwang, wuxi]@fudan.edu.cn

² Universität des Saarlandes,
Campus E 1.4, D-66123 Saarbrücken, Germany
jguo@mmci.uni-saarland.de

³ Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany
[rolf.niedermeier, johannes.uhlmann, mathias.weller]@uni-jena.de

Abstract. Parsimony haplotyping is the problem of finding a smallest size set of haplotypes that can explain a given set of genotypes. The problem is NP-hard, and many heuristic and approximation algorithms as well as polynomial-time solvable special cases have been discovered. We propose improved fixed-parameter tractability results with respect to the parameter “size of the target haplotype set” k by presenting an $O^*(k^{4k})$ -time algorithm. This also applies to the practically important constrained case, where we can only use haplotypes from a given set. Furthermore, we show that the problem becomes polynomial-time solvable if the given set of genotypes is complete, i.e., contains all possible genotypes that can be explained by the set of haplotypes.

1 Introduction

Over the last few years, haplotype inference has become one of the central problems in algorithmic bioinformatics [10,2]. Its applications include drug design, pharmacogenetics, mapping of disease genes, and inference of population histories. One of the major approaches to haplotype inference is *parsimony haplotyping*: Given a set of genotypes, the task is to find a minimum-cardinality set of haplotypes that explains the input set of genotypes. The task to select as few haplotypes as possible (parsimony criterion) is motivated by the observation that in natural populations the number of haplotypes is much smaller than the number of genotypes [2]. Referring for the background in molecular biology to the rich literature (see, e.g., the surveys by Catanzaro and Labbé [2] and Gusfield and Orzack [10]), we focus on the underlying combinatorial problem. In an abstract way, a genotype can be seen as a length- m string over the alphabet $\{0, 1, 2\}$,

* Supported by the DFG, research projects PABI, NI 369/7, and DARE, GU 1023/1, NI 369/11, NSF China (No. 60973026), Shanghai Leading Academic Discipline Project (project number B114), Shanghai Committee of Science and Technology of China (nos. 08DZ2271800 and 09DZ2272800), the Excellence Cluster on Multimodal Computing and Interaction (MMCI), and Robert Bosch Foundation (Science Bridge China 32.5.8003.0040.0).

while a haplotype can be seen as a length- m string over the alphabet $\{0, 1\}$. A set H of haplotypes *explains*, or *resolves*, a set G of genotypes if for every $g \in G$ there is either an $h \in H$ with $g = h$ (trivial case), or there are two haplotypes h_1 and h_2 in H such that, for all $i \in \{1, \dots, m\}$,

- if g has letter 0 or 1 at position i , then both h_1 and h_2 have this letter at position i , and
- if g has letter 2 at position i , then one of h_1 or h_2 has letter 0 at position i while the other one has letter 1.

For example, $H = \{00100, 01110, 10110\}$ resolves $G = \{02120, 20120, 22110\}$. Parsimony haplotyping is NP-hard, and numerous algorithmic approaches based on heuristics and integer linear programming methods are applied in practice [2]. There is also a growing list of combinatorial approaches (with provable performance guarantees) including the identification of polynomial-time solvable special cases, approximation algorithms, and fixed-parameter algorithms [5,13,14,16,11].

In this work, we contribute new combinatorial algorithms for parsimony haplotyping, based on new insights into the combinatorial structure of a haplotype solution. Lancia and Rizzi [14] showed that parsimony haplotyping can be solved in polynomial time if every genotype string contains at most two letters 2, while the problem becomes NP-hard if genotypes may contain three letters 2 [13]. Sharan et al. [16] proved that parsimony haplotyping is APX-hard in even very restricted cases and identified instances with a specific structure that allow for polynomial-time exact solutions or constant-factor approximations. Moreover, they showed that the problem is fixed-parameter tractable with respect to the parameter k = “number of haplotypes in the solution set”. The corresponding exact algorithm has running time $O(k^{k^2+k}m)$. These results were further extended by van Iersel et al. [11] to cases where the *genotype matrix* (the rows are the genotypes and the columns are the m positions in the genotype strings) has restrictions on the number of 2’s in the rows and/or columns. They identified various special cases of haplotyping with polynomial-time exact or approximation algorithms with approximation factors depending on the numbers of 2’s per column and/or row, leaving open the complexity of the case with at most two 2’s per column (and an unbounded number of 2’s per row). Further results in this direction have been recently provided by Cicalese and Milanič [3]. Finally, Fellows et al. [5] introduced the *constrained parsimony haplotyping problem* where the set of haplotypes may not be chosen arbitrarily from $\{0, 1\}^m$ but only from a pool \tilde{H} of plausible haplotypes. Using an intricate dynamic programming algorithm, they extended the fixed-parameter tractability result of Sharan et al. [16] to the constrained case, proving a running time of $k^{O(k^2)} \cdot \text{poly}(m, |\tilde{H}|)$. Jäger et al. [12] recently presented an experimental study of algorithms for computing *all* possible haplotype solutions for a given set of genotypes, where the integer linear programming and branch-and-bound algorithms were sped up using some insights into the combinatorial structure of the haplotype solution, as for example eliminating equal columns from the genotype matrix and recursively decomposing a large problem into smaller ones.

Our contributions are as follows. We simplify and improve the fixed-parameter tractability results of Sharan et al. [16] and Fellows et al. [5] by proposing fixed-parameter algorithms for the constrained and unconstrained versions of parsimony hap-

lotyping that run in $k^{4k} \cdot \text{poly}(m, |\tilde{H}|)$ time, which is a significant exponential speed-up over previous algorithms. Moreover, we develop polynomial-time data reduction rules that yield a problem kernel of size at most $2^k k^2$ for the unconstrained case. A combinatorially demanding part is to show that the problems become polynomial-time solvable if we require that the given set of genotypes is complete in the sense that it contains all genotypes that can be resolved by some pair of haplotypes in the solution set H . We call this special case *induced parsimony haplotyping*, and we distinguish between the case that the genotypes are given as a multiset (note that different pairs of haplotypes may resolve the same genotype), or just as a set without multiplicities. We show that, while there may be an exponential number of optimal solutions in the general case, there can be at most two optimal solutions in the induced case. For both induced cases, unconstrained and constrained, we propose algorithms running in $O(k \cdot m \cdot |G|)$ and $O(k \cdot m \cdot (|G| + |\tilde{H}|))$ time, respectively. Note that these polynomial-time solvable cases stand in sharp contrast to previous polynomial-time solvable cases [3,14,16,11], all of which require a bound on the number of 2's in the genotype matrix.

2 Preliminaries and Definitions

Throughout this paper, we consider *genotypes* as strings of length m over the alphabet $\{0, 1, 2\}$, while *haplotypes* are considered as strings of length m over the alphabet $\{0, 1\}$. If s is a string, then $s[i]$ denotes the letter of s at position i . This applies to both haplotypes and genotypes. Two haplotypes h_1 and h_2 *resolve* a genotype g , denoted by $\text{res}(h_1, h_2) = g$, if, for all positions i , either $h_1[i] = h_2[i] = g[i]$, or $g[i] = 2$ and $h_1[i] \neq h_2[i]$.

For a given set H of haplotypes, let $\text{res}(H) := \{\text{res}(h_1, h_2) \mid h_1, h_2 \in H\}$ denote the set of genotypes resolved by H and $\text{mres}(H)$ the multiset of genotypes resolved by H (the multiplicity of a genotype g in $\text{mres}(H)$ corresponds to the number of pairs of haplotypes in H resolving g). We also write $\text{res}(h, H)$ ($\text{mres}(h, H)$) for the (multi)set of genotypes resolved by h with all haplotypes in H . We say a set H of haplotypes *resolves* a given set G of genotypes if $G \subseteq \text{res}(H)$, and H *induces* G if $\text{res}(H) = G$. If G is a multiset, we similarly require $G \subseteq \text{mres}(H)$ and $\text{mres}(H) = G$, respectively. A haplotype h is *consistent* with a genotype g if $h[i] = g[i]$ for all positions i with $g[i] \neq 2$.

We refer to the monographs [4,6,15] for any details concerning parameterized algorithmics and the survey [9] for an overview on problem kernelization.

We consider the following haplotype inference problems parameterized with the size of the haplotype set H to be computed:

HAPLOTYPE INFERENCE BY PARSIMONY (HIP):

Input: A set G of length- m genotypes and an integer $k \geq 0$.

Question: Is there a set H of length- m haplotypes such that $|H| \leq k$ and $G \subseteq \text{res}(H)$?

In **CONSTRAINED HAPLOTYPE INFERENCE BY PARSIMONY (CHIP)** the input additionally contains a set \tilde{H} of length- m haplotypes and the task is to find a set of at most k haplotypes from \tilde{H} resolving G . Note that with k haplotypes one can resolve at

most $\binom{k}{2} + k$ genotypes. Hence, throughout this paper, we assume that $|G|$ is bounded by $\binom{k}{2} + k$.

In this paper, we introduce the “induced case” of constrained and unconstrained parsimony haplotyping. To simplify the presentation of the results for the induced case, in Section 3 we assume that each genotype contains at least one letter 2. Then, we need two different haplotypes to resolve a genotype. Hence, in the induced case, we assume that $\text{res}(H)$ does not contain an element of H . We claim without proof that our algorithms in Section 3 can be adapted to instances without these restrictions.

Formally, INDUCED (CONSTRAINED) HAPLOTYPE INFERENCE BY PARSIMONY, (C)IHIP for short, is defined as follows. Given a set G of length- m genotypes (and a set \tilde{H} of length- m haplotypes), the task is to find a set $H(\subseteq \tilde{H})$ of length- m haplotypes such that $G = \text{res}(H)$?

Due to the lack of space, some proofs are deferred to a full version of this paper.

3 Induced Haplotype Inference by Parsimony

The main result of this section is that one can solve INDUCED HAPLOTYPE INFERENCE BY PARSIMONY (IHIP) and INDUCED CONSTRAINED HAPLOTYPE INFERENCE BY PARSIMONY (ICHIP) in $O(k \cdot m \cdot |G|)$ and $O(k \cdot m \cdot |G| \cdot |\tilde{H}|)$ time, respectively. In the first paragraph, we consider the following special case of IHIP: given a multiset of $\binom{k}{2}$ length- m genotypes (which are not necessarily distinct), is there a multiset of k length- m haplotypes inducing them? By allowing genotype multisets, we enforce that the input contains information about how often each genotype is resolved by the haplotypes. This allows us to observe a special structure in the input, which makes it easier to present our results. In the second paragraph, we extend our findings to the case that the input genotypes are given as a set, that is, without multiplicities. In this case, we might have some genotypes that are resolved multiple times. However, we do not know in advance which of the input genotypes would be resolved more than once. This makes the set case more delicate than the multiset case. In fact, the set case can be interpreted as a generalization of the multiset case. However, being easier to present, we focus on the multiset case first. Recall that, for the ease of presentation, throughout this section we assume that every genotype contains at least one letter 2 and that $\text{res}(H)$ and $\text{mres}(H)$ do not intersect H .

The Multiset Case. In this paragraph, we show that one can solve INDUCED HAPLOTYPE INFERENCE BY PARSIMONY (IHIP) in $O(k \cdot m \cdot |G|)$ time in the multiset case. This easily generalizes to the constrained case.

We need the following notation. Let $\#_x(i)$ denote the number of genotypes in G which have letter x at position i , for $x \in \{0, 1, 2\}$. We start with a simple structural observation that must be fulfilled by yes-instances. If G is a yes-instance for IHIP, then the set of genotypes restricted to their first positions (i.e., single-letter genotypes) is also a yes-instance. By a simple column-exchange argument, this extends to all positions, implying the following observation (see Fig. 1 for an example).

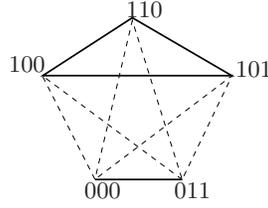


Fig. 1. An example illustrating the Number Condition with $k_0 = 2$ and $k_1 = 3$. Vertices are labeled with haplotypes. Solid edges are genotypes starting with 0 or 1 while dashed edges are genotypes starting with 2.

Observation 1 (“Number Condition”) *If a multiset of genotypes is a yes-instance for IHIP, then, for each position $i \in \{1, \dots, m\}$, there exist two integers $k_0 \geq 0$ and $k_1 \geq 0$ such that $k = k_0 + k_1$, $\#_0(i) = \binom{k_0}{2}$, $\#_1(i) = \binom{k_1}{2}$, and $\#_2(i) = k_0 \cdot k_1$.*

The next lemma is the basis for recursively solving IHIP. For the ease of presentation, we define the operation \oplus . It can be applied to a haplotype h and a genotype g if, for all $i \in \{1, \dots, m\}$, either $h[i] = g[i]$ or $g[i] = 2$. It produces the unique length- m haplotype $h' := h \oplus g$ such that $\text{res}(h, h') = g$. We further define i^* as the first position for which there are genotypes $g, g' \in G$ with $g[i^*] \neq g'[i^*]$. Furthermore, for all $x \in \{0, 1, 2\}$, we denote the set of all genotypes $g \in G$ with $g[i^*] = x$ as G_x . Clearly, any solution for G can be partitioned into a solution for G_0 and a solution for G_1 , as formalized by Lemma 1.

Lemma 1. *Let G be a multiset of genotypes such that not all genotypes in G are identical. Let H be a set of haplotypes inducing G . For $x \in \{0, 1\}$, let H_x denote the haplotypes in H with x at position i^* . Then, H_0 induces G_0 , H_1 induces G_1 , and G_2 is exactly the multiset of genotypes resolved by taking each time one haplotype from H_0 and one haplotype from H_1 . Moreover, $H_0 \cap H_1 = \emptyset$.*

The function $\text{solve}(G)$ (see Alg. 1) recursively computes a solution for G , with the base cases provided by the next two lemmas. Lemma 2 identifies two cases for which there exists a unique solution for G , which in each case can be computed in polynomial time.

Lemma 2. *Assume that $|G| \geq 2$. If all genotypes in G are identical or if $G_x = \emptyset$ for some $x \in \{0, 1\}$, then there exists at most one solution for G . Moreover, in $O(|G| \cdot m)$ time, one can compute a solution or report that G is a no-instance.*

Proof. First we consider the case that all genotypes are identical. Since every genotype has letter 2, Lemma 1 implies that G is a no-instance.

Now, assume that not all genotypes are identical and $G_x = \emptyset$ for some $x \in \{0, 1\}$. Without loss of generality, $G_0 = \emptyset$ and $G_1 \neq \emptyset$. By definition of i^* , $G_2 \neq \emptyset$. Note that in a solution for G there can be at most one haplotype having letter 0 at position i^* (otherwise, we have a contradiction to the fact that $G_0 = \emptyset$). Moreover, there must exist at least one haplotype with 0 at position i^* (otherwise one cannot resolve the haplotypes in G_2). Thus, in any solution H for G , there must exist a unique haplotype $h \in H$

Function solve(G)
Input: A multiset of genotypes $G \subseteq \{0, 1, 2\}^m$.
Output: A set \mathcal{H} containing at most two multisets of haplotypes each of which induces G , if G is a yes-instance; otherwise “no”.

```

1 begin
2   if all genotypes in  $G$  are identical or  $G_x = \emptyset$  for some  $x \in \{0, 1\}$  then
3     return the unique solution  $\{H\}$  (see Lemma 2);
4   else if  $|G_0| = 1$  and  $|G_1| = 1$  then
5     return the at most two solutions  $\{H, H'\}$  (see Lemma 3);
6   else
7     Choose  $x \in \{0, 1\}$  such that  $|G_x| > 1$  and  $|G_x|$  is minimal;
8      $\mathcal{H} \leftarrow \text{solve}(G_x)$ ;
9     forall  $H \in \mathcal{H}$  do replace  $H$  with  $\text{MultisetExtend}(H, G, G_2)$  in  $\mathcal{H}$ ;
10    if  $\mathcal{H}$  contains only the empty set then return “no”;
11    return  $\mathcal{H}$ ;
12  end
13 end

```

Algorithm 1: solve(G) recursively computes all (at most two) solutions for G .

with $h[i^*] = 0$; further, $G_2 = \text{mres}(h, H \setminus \{h\})$. One can now infer all haplotypes as follows. Clearly, one can answer “no” if there is an i , $1 \leq i \leq m$, such that both letters 0 and 1 appear at position i of the genotypes in G_2 . If there is a position i and a $g \in G_2$ with $g[i] \neq 2$, then one can set $h[i] := g[i]$; otherwise, to have a solution for G , all genotypes in G_1 must have the same letter $y \in \{0, 1\}$ at this position, so one can set $h[i] := 1 - y$. With h settled, one can easily determine the haplotypes h' with $h'[i^*] = 1$ (these are the haplotypes $g \oplus h$ for $g \in G_2$). Finally, one has to make sure that all these haplotypes induce G . If not, then the input instance is a no-instance. The running time of this procedure is $O(|G| \cdot m)$. \square

Next, we show that there are at most two solutions for G if each of G_0 and G_1 contains only a single genotype.

Lemma 3. *If $|G_0| = 1$ and $|G_1| = 1$, then there are at most two solutions for G . Moreover, in $O(m)$ time, one can compute these solutions or report that G is a no-instance.*

Proof. Let g_0 and g_1 be the genotypes in G_0 and G_1 , respectively. By Lemma 1, two pairs of haplotypes are required to resolve them, denoted by h_0^0 and h_0^1 (resolving g_0), and h_1^0 and h_1^1 (resolving g_1). If $|G_2| \neq 4$, then return “no” (see Observation 1); otherwise, let $G_2 = \{g_2, g_3, g_4, g_5\}$. If none of g_0 and g_1 contains letter 2, then the haplotypes are easily constructed (they are equal to the respective genotype). Otherwise, let i be the first position where g_0 or g_1 has letter 2, say $g_0[i] = 2$. Without loss of generality, let $h_0^0[i] := 0$ and $h_0^1[i] := 1$. We consider the following two cases:

Case 1: $g_1[i] \neq 2$.

Without loss of generality, let $g_1[i] = 0$. Then, two of the genotypes in G_2 must have 0 at position i and the other two must have 2 at position i ; otherwise, return “no”. Without loss of generality, let $g_2[i] = g_3[i] = 0$ and $g_4[i] = g_5[i] = 2$. Since g_2 and g_3 must

be resolved by h_0^0 , one can uniquely determine h_0^0 as follows. Consider any position j . If $g_2[j] \neq 2$ and $g_3[j] \neq 2$, then they must both be equal (if not, then return “no”). In this case, let $h_0^0[j] = g_2[j]$. If exactly one of $g_2[j]$ and $g_3[j]$ is equal to 2, say $g_3[j] = 2$, then let $h_0^0[j] = g_2[j]$. If $g_2[j] = g_3[j] = 2$, then we know that $g_1[j] \neq 2$ (otherwise, return “no”) and thus $h_0^0[j] := 1 - g_1[j]$. Finally, let $h_0^1 := h_0^0 \oplus g_0$, $h_1^0 := h_0^0 \oplus g_2$, and $h_1^1 := h_0^0 \oplus g_3$. If these haplotypes also correctly resolve g_1 , g_4 , and g_5 , then we have a unique solution for G , otherwise return “no”.

Case 2: $g_0[i] = g_1[i] = 2$.

There is a genotype in G_2 having 0 at position i and another having 1 at position i (otherwise, return “no”). Without loss of generality, let $g_2[i] = 0$, $g_3[i] = 1$, $h_1^0[i] := 0$, and $h_1^1[i] := 1$. Then, $g_4[i] = g_5[i] = 2$ and $g_2 = \text{res}(h_0^0, h_1^0)$ and $g_3 = \text{res}(h_0^0, h_1^1)$. Now there are two possibilities to resolve g_4 and g_5 . Either $g_4 = \text{res}(h_0^1, h_1^0)$ and $g_5 = \text{res}(h_0^0, h_1^1)$, or $g_4 = \text{res}(h_0^0, h_1^1)$ and $g_5 = \text{res}(h_0^1, h_1^0)$. By choosing one of these two possibilities, all four haplotypes are fixed. Thus, there are at most two solutions for G .

Note that there are only six genotypes. Thus, for every position the computations are clearly doable in constant time. Hence, the whole procedure runs in $O(m)$ time. \square

The next two lemmas show that one can solve an IHIP instance recursively if neither Lemma 2 nor Lemma 3 applies. That is, we now assume that not all genotypes are identical and we have $|G_x| > 1$ for some $x \in \{0, 1\}$. We show that, given a solution for G_x , one can uniquely extend this solution to a solution for G , or decide that G is a no-instance, leading to function `MultiSetExtend` (see Alg. 2)

Lemma 4. *Let $|G_x| > 1$ for some $x \in \{0, 1\}$, let H_x be a multiset of haplotypes inducing G_x , and let g be a genotype in G_2 with the smallest number of 2’s. If G is induced by H with $H_x \subseteq H$, then all haplotypes in H_x consistent with g must be identical.*

Proof. Without loss of generality, we assume that $|G_0| > 1$. Suppose that there is an H with $H_x \subseteq H$ inducing G . Since $g[i^*] = 2$, there must be a haplotype $h_1 \in H_x$ and a haplotype $h_2 \in H \setminus H_x$ resolving g . Clearly, h_1 and h_2 are consistent with g . We show that there is no other haplotype $h \in H_x$ such that $h \neq h_1$ and h is consistent with g . For the sake of contradiction, assume that there is such a haplotype h . First, note that h , h_1 , and h_2 are consistent with g and hence identical at positions where g does not have letter 2. Since $h \neq h_1$, h differs from h_1 in at least one of the positions where g has letter 2. Thus, h_2 (which together with h_1 resolves g and hence is the complement of h_1 at the positions where g has letter 2) must have the same letter as h at some position where h_1 and h_2 differ. This implies that $\text{res}(h, h_2) \in G_2$ has fewer 2’s than g , contradicting the choice of g . \square

Lemma 5. *Let $|G_x| > 1$ for some $x \in \{0, 1\}$, and let H_x be a multiset of haplotypes inducing G_x . If G is induced by H with $H_x \subseteq H$, then H is uniquely determined and function `MultiSetExtend` (see Alg. 2) computes H in $O(|H_x| \cdot |G_2| \cdot m)$ time.*

Proof. The correctness of lines 4–7 of `MultiSetExtend` (see Alg. 2) follows from Lemma 4. Since including $h' := h \oplus g$ in H is the only choice, the genotypes resolved by h' and other haplotypes in H_x should also be in G_2 ; otherwise, no solution exists.

Function `MultisetExtend`(H_x, G, G_2)
Input: A haplotype multiset H_x inducing G_x for some $x \in \{0, 1\}$, and a multiset G_2 of genotypes.
Output: A haplotype multiset H inducing G with $H_x \subseteq H$, if one exists; otherwise an empty set.

```

1 begin
2    $H := H_x$ ;
3   while  $G_2 \neq \emptyset$  do
4     Choose a  $g \in G_2$  with smallest number of 2's;
5     Choose an  $h \in H_x$  consistent with  $g$ ;
6      $h' := h \oplus g$ ;
7      $H := H \cup \{h'\}$ ;
8      $G' := \{g' \mid \exists h'' \in H_x : g' = \text{res}(h', h'')\}$ ;
9     if  $G' \not\subseteq G_2$  then return  $\emptyset$ ;
10     $G_2 := G_2 \setminus G'$ ;
11  end
12  if  $\text{mres}(H) = G$  then return  $H$ ;
13  else return  $\emptyset$ ;
14 end

```

Algorithm 2: An algorithm to extend a solution for G_x to G in the multiset case.

Thus, lines 8 and 9 of `MultisetExtend` are correct. Line 10 of `MultisetExtend` safely removes the genotypes resolved by h' from G_2 . The next while-iteration proceeds to find the next pair consisting of a haplotype h and a genotype $g \in G_2$ satisfying Lemma 4. If there is a solution for G comprising H_x , then we must end up with an empty G_2 . Moreover, $H \setminus H_x$ should resolve all genotypes in G_{1-x} and, together with H_x , the genotypes in G_2 ; this is examined in line 12 of `MultisetExtend`. Thus, the function `MultisetExtend` is correct. By Lemma 4, the solution H with $H_x \subseteq H$ is unique.

Concerning the running time, note that the most time-consuming part of the function is to find the consistent haplotypes in H_x for a given genotype in G_2 . This can be done in $O(|H_x| \cdot |G_2| \cdot m)$ time by iterating over all haplotypes in H_x and for each haplotype over all genotypes in G_2 . \square

Putting all together, we obtain the main theorem of this paragraph.

Theorem 1. *In case of a multiset G of length- m genotypes, INDUCED HAPLOTYPE INFERENCE BY PARSIMONY and CONSTRAINED INDUCED HAPLOTYPE INFERENCE BY PARSIMONY can be solved in $O(k \cdot m \cdot |G|)$ and $O(k \cdot m \cdot (|G| + |\tilde{H}|))$ time, respectively.*

Proof. (Sketch) We show that the algorithm `solve`(G) (see Alg. 1) is correct. If all genotypes are identical or $G_x = \emptyset$, for some $x \in \{0, 1\}$, then the correctness follows from Lemma 2. Hence, in the following, assume that not all genotypes are identical, $G_0 \neq \emptyset$, and $G_1 \neq \emptyset$. Distinguish the cases that $|G_0| = |G_1| = 1$ and $|G_x| > 1$, for some $x \in \{0, 1\}$. In the case that $|G_0| = |G_1| = 1$, one can compute the solutions (at most two) for G using Lemma 3. In the other case, for some $x \in \{0, 1\}$, it

holds that $|G_x| > 1$ and $|G_{1-x}| > 0$. Without loss of generality, assume $|G_0| > 1$. By Lemma 1, a solution for G consists of a solution H_0 for G_0 and a solution H_1 for G_1 , and $H_0 \cap H_1 = \emptyset$. Since one tries to extend every solution for G_0 and these extensions are unique by Lemma 5, one will find every possible solution for G . Since the base cases have at most two solutions and extensions are uniquely determined by Lemma 5, there exist at most two solutions for G . In the constrained case, one only needs to check whether one of the computed solutions is in the given set of haplotypes. The claimed running time follows from Lemmas 2, 3, and 5. \square

The Set Case. If the input is not a multiset, but a set G of genotypes, that is, all genotypes in G are pairwise distinct, then the Number Condition (Observation 1) does not necessarily hold. Consider the haplotype set $H = \{000, 001, 110, 111\}$ which induces the set $\text{res}(H) = \{002, 112, 221, 220, 222\}$, but also induces the multiset $\text{mres}(H) = \{002, 112, 221, 220, 222, 222\}$ (observe that $\text{res}(000, 111) = \text{res}(001, 110) = 222$). The problem is that we cannot directly infer from G which genotypes should be resolved more than once. However, many properties of the multiset case (as for example Lemmas 1, 2, and 3) carry over to the set case, so we only need a moderate modification of the multiset algorithm to solve the set case. More specifically, the key to solve the set case is to adapt function `MultisetExtend` (all details are deferred to the long version of this paper).

Theorem 2. *In case of a set G of length- m genotypes, INDUCED HAPLOTYPE INFERENCE BY PARSIMONY and CONSTRAINED INDUCED HAPLOTYPE INFERENCE BY PARSIMONY can be solved in $O(k \cdot m \cdot |G|)$ and $O(k \cdot m \cdot (|G| + |\tilde{H}|))$ time, respectively.*

4 General Haplotype Inference by Parsimony

This section contains an algorithm to solve the general parsimony haplotyping problem for the unconstrained and the constrained versions in $O(k^{4k+1} \cdot m)$ and $O(k^{4k+1} \cdot m \cdot |\tilde{H}|)$ time, respectively, improving and partially simplifying previous fixed-parameter tractability results [16,5]. In addition, we provide a simple kernelization.

We start with some preliminary considerations. Given a set of haplotypes resolving a given set of genotypes, the relation between the haplotypes and the genotypes can be depicted by an undirected graph, the *solution graph*, in which the edges are labeled by the genotypes and every vertex v is labeled by a haplotype h_v . If an edge $\{u, v\}$ is labeled by genotype g , we require that $g = \text{res}(h_u, h_v)$. We call such a vertex/edge labeling *consistent*. If only the edges are labeled, the graph is an *inference graph* (because it allows us to infer all the haplotypes). Solution graphs and inference graphs may contain loops.

In what follows, assume that the input is a yes-instance, i.e., a solution graph exists. Intuitively, our algorithm “guesses” an inference graph for G (by enumerating all possible such graphs) and then infers the haplotypes from the genotype labels on the edges. To this end, it guesses for every connected component of the solution graph a spanning subgraph with edges labeled by some of the genotypes in G in such a way

Input: A set of genotypes $G \subseteq \{0, 1, 2\}^m$ and an integer $k \geq 0$.

Output: Either a set of haplotypes H with $|H| \leq k$ and $G \subseteq \text{res}(H)$, or “no” if there is no solution of size at most k .

```

1 forall size- $k$  subsets  $G' \subseteq G$  do
2   forall inference graphs  $\Gamma$  for  $G'$  on  $k$  vertices and  $k$  edges do
3     forall non-bipartite connected components of  $\Gamma$  do
4       if possible, compute the labels of all vertices of the component (Lemma 7),
5       otherwise try the next inference graph (goto line 2);
6     end
7     forall bipartite connected components of  $\Gamma$  do
8       if possible, compute a consistent vertex labeling for the component
9       (Lemma 8), otherwise try the next inference graph (goto line 2);
10    end
11    Let  $H$  denote the inferred haplotypes (vertex labels);
12    if  $G \subseteq \text{res}(H)$  then return  $H$ ;
13  end
14 return “no”;

```

Algorithm 3: An algorithm solving HIP in $O(k^{4k+1} \cdot m)$ time.

that we have enough information at hand to infer the haplotypes. Then, one has to solve the following subproblem: Given an inference graph for a subset of genotypes of G , does there exist a consistent vertex labeling? The next three lemmas show how to solve this subproblem by separately considering the connected components of the inference graphs.

Lemma 6. *Let G be a set of genotypes and let $\Gamma = (V, E)$ be a connected inference graph for G . For each position i , $1 \leq i \leq m$, if there is a genotype $g \in G$ with $g[i] \neq 2$, then one can, in $O(|V| + |E|)$ time, uniquely infer the letters of all haplotypes at position i or report that there is no consistent vertex labeling.*

Lemma 7. *Let $\Gamma = (V, E)$ be a connected inference graph for a set G of genotypes that contains an odd-length cycle. Then, there exists at most one consistent vertex labeling. Furthermore, one can compute in $O(m \cdot (|V| + |E|))$ time a consistent vertex labeling or report that no consistent vertex labeling exists.*

Lemma 8. *Let $\Gamma = (V_a, V_b, E)$ be a connected bipartite inference graph for a set G of length- m genotypes. Let $u \in V_a$ and $w \in V_b$ be arbitrarily chosen. Then,*

1. *one can compute in $O(m \cdot (|V_a| + |V_b| + |E|))$ time a consistent vertex labeling or report that no consistent vertex labeling exists, and*
2. *the genotypes resolved by h_u and h_w are identical for every consistent vertex labeling.*

Next, we describe the algorithm for the unconstrained version (HIP), see Alg. 3. To solve HIP, we could enumerate all inference graphs for G and then find the vertex labeling using Lemmas 7 and 8. However, to be more efficient, we first select a size- k subset of genotypes (line 1 of Alg. 3), and then we enumerate all inference graphs

on k vertices containing exactly k edges labeled by the k chosen genotypes (line 2 of Alg. 3). Assume that there exists a solution graph for G . Of all inference graphs on k vertices and k edges consider one with the following properties:

- it contains a spanning subgraph of every connected component of the solution graph, and
- the spanning subgraph of any non-bipartite connected component contains an odd cycle (thus, the bipartite components of the inference graph are exactly the bipartite components of the solution graph).

Obviously, this inference graph exists and is considered by Alg. 3. By Lemma 7, we can uniquely infer the vertex labels for all connected components of the inference graph containing an odd cycle. For every bipartite component, we can get a consistent vertex labeling from Lemma 8. In such a bipartite component, for any two vertices $u \in V_a$ and $v \in V_b$, the genotypes resolved by h_u and h_v are identical for every consistent vertex labeling. Thus, the haplotypes resolve all genotypes contained in the respective (bipartite) component of the solution graph. In summary, if the given instance is a yes-instance, then our algorithm will find a set of at most k haplotypes resolving the given genotypes.

Theorem 3. HAPLOTYPE INFERENCE BY PARSIMONY and CONSTRAINED HAPLOTYPE INFERENCE BY PARSIMONY can be solved in $O(k^{4k+1} \cdot m)$ and $O(k^{4k+1} \cdot m \cdot |\tilde{H}|)$ time, respectively.

Proof. (Sketch) We first consider the unconstrained case. By the discussion above, Alg. 3 correctly solves HIP. It remains to analyze its running time. First, there are $O(\binom{|G|}{k})$ size- k subsets G' of G . Second, there are $O(k^{2k})$ inference graphs on k vertices containing exactly k edges labeled by the genotypes in G' because for every genotype $g \in G'$ we have k^2 choices for the endpoints (loops are allowed) of the edge labeled by g . For each of those inference graphs, applying Lemma 7 and Lemma 8 to its connected components takes $O(k \cdot m)$ time. Hence, the overall running time of Alg. 3 sums up to $O(\binom{|G|}{k} \cdot k^{2k} \cdot m \cdot k)$. Since $|G| \leq k^2$, the running time can be bounded by $O(k^{4k+1} \cdot m)$.

One can easily adapt Alg. 3 to solve CHIP as follows. As before, one enumerates all size- k subsets $G' \subseteq G$ and all inference graphs for G' . Since, by Lemma 7, the vertex labels for the connected components containing an odd cycle are uniquely determined, one only has to check whether the inferred haplotypes are contained in the given haplotype pool \tilde{H} (otherwise, try the next inference graph). Basically, the only difference is how to proceed with the bipartite components of the inference graph. Let (W, F) be a connected bipartite component of the current inference graph. Instead of choosing an arbitrary consistent vertex labeling as done in Lemma 8, proceed as follows. Choose an arbitrary vertex $v \in W$ and check for every haplotype $h \in \tilde{H}$ whether there exists a consistent vertex labeling for this component where v is labeled by h . Note that fixing the vertex label for v implies the existence of at most one consistent vertex labeling of (W, F) . If it exists, this labeling can be computed by a depth-first traversal starting at v . If for a haplotype h there exists a consistent vertex labeling of (W, F) such that all labels are contained in \tilde{H} , then proceed with the next bipartite component. Otherwise, one can conclude that for the current inference graph there is no consistent vertex

labeling using only the given haplotypes, and, hence, one can proceed with the next inference graph. The correctness and the claimed running time follow by almost the same arguments as in the unconstrained case. \square

Problem Kernelization. In this paragraph, we show that HIP admits an exponential-size problem kernel. To this end, we assume the input G to be in the matrix representation that is mentioned in the introduction; that is, each row represents a genotype while each column represents a position. Since it is obvious that we can upper-bound the number n of genotypes in the input by k^2 , it remains to bound the number m of columns (positions) in the input. The idea behind the following data reduction rule is that we can safely delete a column if there is another column that is identical. By applying this rule exhaustively, we can bound the number of columns by 2^k .

Reduction Rule. *Let (G, k) be an instance of HIP. If two columns of G are equal, then delete one of them.*

The correctness of the reduction rule follows by the observation that, given at most k haplotypes resolving the genotypes in the reduced instance, we can easily find a solution for the original instance by copying the respective haplotype positions. Next, we bound the number of columns.

Lemma 9. *Let (G, k) be a yes-instance of HIP that is reduced with respect to the reduction rule. Then, G has at most 2^k columns.*

Proof. Let H denote a matrix of k haplotypes resolving G . It is obvious that if two columns i and j of H are equal, then columns i and j of G are equal. Now, since G does not contain a pair of equal columns, neither does H . Since there are only 2^k different strings in $\{0, 1\}^k$, it is clear that H cannot contain more than 2^k columns and thus, neither can G . \square

Since the number n of genotypes can be bounded by k^2 and the number m of columns can be bounded by 2^k (Lemma 9), one directly obtains Proposition 1.

Proposition 1. *HAPLOTYPE INFERENCE BY PARSIMONY admits a problem kernel of size at most $2^k \cdot k^2$ that can be constructed in $O(n \cdot m \cdot \log m)$ time.*

Plugging Proposition 1 into Theorem 3, we achieve the following.

Corollary 1. *HIP can be solved in $O(k^{4k+1} \cdot 2^k + n \cdot m \cdot \log m)$ time.*

5 Conclusion

We contributed new combinatorial algorithms for parsimony haplotyping with the potential to make the problem more feasible in practice without giving up the demand for optimal solutions. Our results also lead to several new questions for future research. For instance, our kernelization result yields a problem kernel of exponential size. It would be interesting to know whether a polynomial-size problem kernel exists, which may

also be seen in the light of recent breakthrough results on methods to prove the non-existence of polynomial-size kernels [1,7]. A second line of research is to make use of the polynomial-time solvable induced cases to pursue a “distance from triviality” approach [8]. The idea here is to identify and exploit parameters that measure the distance of general instances of parsimony haplotyping to the “trivial” (that is, polynomial-time solvable) induced cases. Research in this direction is underway. A more speculative research direction could be to investigate whether our results on the induced case (with at most two optimal solutions) may be useful in the context of recent research [12] on finding all optimal solutions in the general case. Clearly, it remains an interesting open problem to find a fixed-parameter algorithm for parsimony haplotyping with an exponential factor of the form c^k for some constant c .

References

1. H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. Comput. System Sci.*, 75(8):423–434, 2009. 13
2. D. Catanzaro and M. Labbé. The pure parsimony haplotyping problem: Overview and computational advances. *International Transactions in Operational Research*, 16(5):561–584, 2009. 1, 2
3. F. Cicalese and M. Milanič. On parsimony haplotyping. Technical Report 2008-04, Universität Bielefeld, 2008. 2, 3
4. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999. 3
5. M. R. Fellows, T. Hartman, D. Hermelin, G. M. Landau, F. A. Rosamond, and L. Rozenberg. Haplotype inference constrained by plausible haplotype data. In *Proc. 20th CPM*, volume 5577 of *LNCS*, pages 339–352. Springer, 2009. 2, 9
6. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006. 3
7. L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *Proc. 40th STOC*, pages 133–142. ACM Press, 2008. 13
8. J. Guo, F. Hüffner, and R. Niedermeier. A structural view on parameterizing problems: Distance from triviality. In *Proc. 1st IWPEC*, volume 3162 of *LNCS*, pages 162–173. Springer, 2004. 13
9. J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007. 3
10. D. Gusfield and S. H. Orzack. Haplotype inference. *CRC Handbook on Bioinformatics*, chapter 1, pages 1–25. CRC Press, 2005. 1
11. L. van Iersel, J. Keijsper, S. Kelk, and L. Stougie. Shorelines of islands of tractability: Algorithms for parsimony and minimum perfect phylogeny haplotyping problems. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 5(2):301–312, 2008. 2, 3
12. G. Jäger, S. Climer, and W. Zhang. Complete parsimony haplotype inference problem and algorithms. In *Proc. 17th ESA*, volume 5757 of *LNCS*, pages 337–348. Springer, 2009. 2, 13
13. G. Lancia, M. C. Pinotti, and R. Rizzi. Haplotyping populations by pure parsimony: Complexity of exact and approximation algorithms. *INFORMS Journal on Computing*, 16(4):348–359, 2004. 2
14. G. Lancia and R. Rizzi. A polynomial case of the parsimony haplotyping problem. *Operations Research Letters*, 34:289–295, 2006. 2, 3
15. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. 3
16. R. Sharan, B. V. Halldórsson, and S. Istrail. Islands of tractability for parsimony haplotyping. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 3(3):303–311, 2006. 2, 3, 9