INFORMS Journal on Computing
Vol. 16, No. 4, Fall 2004, pp. 348–359
ISSN 0899-1499 | EISSN 1526-5528 | 04 | 1604 | 0348

DOI 10.1287/ijoc.1040.0085
© 2004 INFORMS

Haplotyping Populations by Pure Parsimony: Complexity of Exact and Approximation Algorithms

Giuseppe Lancia
Dipartimento di Matematica e Informatica, Università di Udine, Via delle Scienze 206, 33100 Udine, Italy, lancia@dimi.uniud.it

Maria Cristina Pinotti
Dipartimento di Matematica e Informatica, Università di Perugia, Via Vanvitelli 1, 06123 Perugia, Italy, pinotti@unipg.it

Romeo Rizzi
Dipartimento di Informatica e Telecomunicazioni, Università di Trento, Via Sommarive 14, 38050 Povo (TN), Italy, romeo@science.unitn.it

In this paper we address the *pure parsimony haplotyping* problem: Find a minimum number of haplotypes that explains a given set of genotypes. We prove that the problem is APX-hard and present a 2^{k-1}-approximation algorithm for the case in which each genotype has at most k ambiguous positions. We further give a new integer-programming formulation that has (for the first time) a polynomial number variables and constraints. Finally, we give approximation algorithms, not based on linear programming, whose running times are almost linear in the input size.

Key words: pure parsimony haplotyping; SNPs; deterministic rounding; node cover; compact integer program
History: Accepted by Harvey J. Greenberg, Guest Editor; received August 2003; revised October 2003, February 2004; accepted March 2004.

1. Introduction

A *single nucleotide polymorphism* (SNP) is a site of the human genome (i.e., the position of a specific nucleotide) whose content shows a statistically significant variability within a population. A position is considered a SNP if for the minority of the population (as long as it consists of at least 5% of the population) a certain nucleotide is observed (called the least frequent *allele*) while for the rest of the population a different nucleotide is observed (the most frequent allele).

The recent completion of the sequencing phase of the Human Genome Project (Venter et al. 2001, International Human Genome Sequencing Consortium 2001) has shown that the genomes of two different individuals are identical in about 99% of the positions, and that most polymorphisms (i.e., differences at genomic level) are in fact SNPs, occurring, on average, every thousand bases (Chakravarti 1998).

Because SNPs are the predominant form of human variation, their importance can hardly be overestimated, and they are widely used in therapeutic, diagnostic, and forensic applications. There is a large amount of research going on in determining SNP sites in humans as well as other species, with a SNP consortium founded with the aim of designing a detailed SNP map for the human genome (Marshall 1999, Helmuth 2001).

Humans are *diploid* organisms, i.e., their DNA is organized in pairs of chromosomes. For each pair of chromosomes, one chromosome copy is inherited from the father and the other copy is inherited from the mother. For a given SNP, an individual can be either *homozygous* (i.e., possess the same allele on both chromosomes) or *heterozygous* (i.e., possess two different alleles). The values of a set of SNPs on a particular chromosome copy define a *haplotype*. *Haplotyping* an individual consists of determining a pair of haplotypes, one for each copy of a given chromosome. The pair provides full information of the SNP fingerprint for that individual at the specific chromosome.

With the larger availability in SNP genomic data, recent years have seen the birth of a set of new combinatorial and optimization problems related to SNPs. In particular, because it is impractical to perform the complete sequencing of an individual's genome as a routine experiment, most combinatorial problems are related to haplotyping individuals without sequencing their genomes (however, even in the case of a fully sequenced genome, unavoidable errors in the data lead to the definition of mathematical haplotyping problems such as the *single individual haplotyping problem* (Lancia et al. 2001, Lippert et al. 2002, Rizzi et al. 2002, Li et al. 2003)).

The cheapest way to haplotype a population (i.e., a set of individuals), is first to obtain ambiguous *genotype* data, and then retrieve the haplotypes computationally. Genotype data provide, for each individual, information about the multiplicity of each SNP allele; i.e., for each SNP site and each individual of a population, it is known if the individual is homo- or heterozygous. The ambiguity comes from heterozygous sites, because, to retrieve the haplotypes, one has to decide how to distribute the two allele values on the two chromosome copies. *Resolving* (or *explaining*) a genotype g requires determining two haplotypes such that, if they are assumed to be the two chromosome copies, then, computing the multiplicity of each SNP allele, we obtain exactly the genotype g. The *population haplotyping problem* is, given a set $\mathcal{G}$ of genotypes, determine a set $\mathcal{H}$ of haplotypes such that each genotype $g \in \mathcal{G}$ is explained by two haplotypes $h', h'' \in \mathcal{H}$. For its importance (as we said, haplotyping from genotype data is nowadays the only viable way) the population haplotyping problem has been and is being extensively studied, under many objective functions, among which:

- *Perfect phylogeny*: Under this model, the solution (i.e., the set $\mathcal{H}$ of haplotypes computed from the input genotype data) must fit a perfect phylogeny. That is, there must exist a tree in which (i) the set of leaves are the haplotypes; (ii) each SNP labels an edge; and (iii) the partition of $\mathcal{H}$ induced by the removal of an edge labeled by a SNP s is such that haplotypes in the same subset of the partition have the same allele value at s. The perfect phylogeny haplotyping problem was shown to be polynomial (Bafna et al. 2003, Eskin et al. 2003).
- *Clark's rule*: In this version of the problem, the solution $\mathcal{H}$ is required to be obtained via successive applications of a rule known as *Clark's rule*. Biologist A. Clark (1990) suggested a greedy rule to resolve genotypes. This rule starts from a minimal "bootstrap" set of haplotypes and uses them to explain as many genotypes as possible, while greedily introducing new haplotypes when needed. Gusfield studied the problem of how to apply Clark's rule in an optimal way, showing this problem to be APX-hard and suggesting an integer-programming formulation for its solution (Gusfield 2000, 2001).
- *Pure parsimony*: Under this model, it is required that $\mathcal{H}$ has the minimum possible cardinality. The objective is based on the principle that, under many explanations of an observed phenomenon, one should choose the one that requires the fewest assumptions. Here, one is trying to determine what is the minimum number of different elements (haplotypes) that, combined in pairs during time, have given rise to a set of observed diversities (genotypes). This problem has been studied by Gusfield (2003), who adopted an integer-programming formulation for its practical solution. The problem is NP-hard, as first shown by Hubbel (2002).

Each model and objective function has specific biological motivations, which are discussed in the cited references. In this paper, we focus on the pure parsimony haplotyping problem. Haplotyping by pure parsimony is the most recent model, whose importance is now being recognized as crucial in the solution of more complex haplotyping problems. In fact, when observed at the largest possible scale (e.g., considering several thousand SNPs at once) haplotypes are not inherited as units, but there is a certain level of recombination (i.e., new haplotypes are created by merging pieces of other haplotypes). A *block* is a region where no recombination has occurred in any of the haplotypes. Biological reasons suggest that the number of different haplotypes observed within a block should be minimum (Gusfield 2003).

1.1. Our Results and Paper Organization

The problem we study is the following: Find a set $\mathcal{H}$ of haplotypes of *smallest* cardinality that explains a given set $\mathcal{G}$ of m genotypes. The binary nature of SNPs leads to a nice combinatorial problem. As described below, this problem is defined over a matrix with entries in $\{0, 1, 2\}$.

We first address the complexity of the problem, and prove that the problem is not only NP-hard, but in fact is APX-hard. That is, there is a constant $\lambda > 1$ such that the existence of a λ-approximation algorithm for the problem would imply P = NP. Our reduction is from the NODE-COVER problem and is described in §3.

Because of its complexity, any exact algorithm for the problem must resort to some sort of enumeration. In §4 we describe two integer linear-programming (ILP) formulations of the problem. The first formulation, described in §4.1., has an exponential number of variables and constraints, and was used in Gusfield (2003) to obtain a practical solution for problem instances of moderate (but still important for real-life applications) size. Later in the paper we use this formulation to obtain an approximation algorithm based on linear programming. The second ILP formulation, described in §4.2., is the first formulation for this problem with a polynomial number of variables and constraints.

In the third part of the paper, we address the issue of approximation algorithms for this problem. After showing that it is easy to obtain a $\sqrt{m}$-approximation, we obtain better approximation algorithms whose ratio depends on the maximum number, k, of heterozygous sites of any input genotype. In particular, by using the ILP formulation of §4.1., we describe in §5 a 2^{k-1}-approximation algorithm based on deterministic rounding of an optimal linear-programming solution.

By using a different approach we are able to obtain a similar result. The approach, based on random sampling, does not require the use of an LP solver. In §6.2. we describe a (Monte Carlo) randomized polynomial algorithm, that almost surely (i.e., with probability at least $(m-1)/m$) returns a feasible solution $\mathcal{H}_{feas}$ with $|\mathcal{H}_{feas}| \leq 2^{k+1}\lfloor \log m \rfloor (1+\lceil \ln m \rceil)$OPT. The running time of the algorithm is bounded by a polynomial in the size of the input instance. More precisely, where $m = |\mathcal{G}|$ and n is the number of SNPs, then the algorithm runs in $O(m \log^3 m(n + \log m))$ time. Hence, this Monte Carlo approach is extremely effective in that it leads to a simple-to-implement almost-linear-time algorithm. Despite its simplicity, the theoretical analysis and motivation of the algorithm is involved. To introduce and motivate our Monte Carlo algorithm better, we first design a similar randomized (Las Vegas) algorithm, described in §6.1., that assumes the ability to know the OPT value and constructs a feasible solution $\mathcal{H}_{feas}$ with $|\mathcal{H}_{feas}| \leq 2^{k+1}\lfloor \log m \rfloor (1+\lceil \ln m \rceil)$OPT. The running time of the algorithm is a random variable with expected value bounded by a polynomial in the size of the input instance.

2. Preliminaries: SNPs, Haplotypes, and Genotypes

A single nucleotide polymorphism, or SNP, is a position in the genome at which some of us have one base while the others have a different base. The two base values are called *alleles*. Due to the binary nature of SNPs, we encode for each SNP the two alleles with the bits 0 and 1. Diploid genomes (such as the human genome) are organized into pairs of chromosomes (a paternal and a maternal copy) that have nearly identical content and carry (paternal and maternal) copies of the same genes. For each SNP, an individual is homozygous if the SNP has the same allele on both chromosome copies, and otherwise the individual is heterozygous. The values of a set of SNPs on a particular chromosome copy define a haplotype. In Figure 1 we give a simplistic example of a chromosome with three SNP sites. The individual is heterozygous at SNPs 1 and 3, and homozygous at SNP 2. The haplotypes are CCA and GCT. Under an encoding of the alleles, these two haplotypes could be represented as the binary vectors $(0, 1, 0)$ and $(1, 1, 1)$. In this encoding, a "0" at SNP 1 stands for C and a "1" stands for G.

```
Chrom. c, paternal:  ataggtccCtatttccaggcgcCgtatacttcgacgggActata
Chrom. c, maternal:  ataggtccGtatttccaggcgcCgtatacttcgacgggTctata

Haplotype 1 →                C             C               A
Haplotype 2 →                G             C               T
```

Figure 1 **A Chromosome and the Two Haplotypes**

Haplotype data are particularly sought after in the study of complex diseases (those affected by more than a single gene) since they can give complete information about the set of gene alleles that are inherited. However, polymorphism screens are conducted on large populations where it is not feasible to examine the two copies of each chromosome separately, and genotype data rather than haplotype data are usually obtained. A genotype describes the multiplicity of each SNP allele for the chromosome of interest. At each SNP, three possibilities arise: Either one is homozygous for the allele 0, or homozygous for the allele 1, or heterozygous. A *genotype* can then be represented with a *vector with entries in* $\{0, 1, 2\}$, where each position with a 0 or 1 corresponds to a homozygous site for the allele 0 or 1, respectively, and each position with a 2 (called an *ambiguous* position) corresponds to a heterozygous site. A *haplotype* is a *vector with entries in* $\{0, 1\}$ (i.e., a binary vector). In what follows, n denotes the length of a genotype and a haplotype vector.

2.1. Problem Definition

For any vector v, we denote by $v[i]$ the ith component of v.

When h_1 and h_2 are haplotypes, their *sum* $g = h_1 \oplus h_2$ is a genotype and is defined as follows:

$$g[i] = \begin{cases} h_1[i] & \text{if } h_1[i] = h_2[i] \\ 2 & \text{otherwise} \end{cases} \qquad (\text{for } i = 1, \ldots, n).$$

We say that a genotype g is *resolved* (or *explained*) by a pair of haplotypes $\{h, q\}$ if $g = h \oplus q$. A haplotype h is called *compatible* with a genotype g if h agrees with g at all nonambiguous positions.

The problem input is a *population*, i.e., a family $\mathcal{G} = \{g_1, \ldots, g_m\}$ of m genotypes, on n SNPs. The input is viewed as an $m \times n$ matrix M with entries in $\{0, 1, 2\}$. Each row of M is a genotype of $\mathcal{G}$. This interpretation of the input is useful in formalizing the integer-programming model of the problem described in §4.1.

We say that a set $\mathcal{H}$ of haplotypes *explains* $\mathcal{G}$ if for every $g \in \mathcal{G}$ there exist $h_1, h_2 \in \mathcal{H}$ such that $g = h_1 \oplus h_2$. The output of the problem consists of a set $\mathcal{H}$ of haplotypes that explains $\mathcal{G}$. The output is seen as a $2m \times n$ binary matrix, in which each row is a haplotype and there is a one-to-one correspondence of rows of M and pairs of rows of M' (i.e., each row g of M is "expanded" into two rows g', g'' of M' such that $g' \oplus g'' = g$. Note that if g is nonambiguous, the two rows g' and g'' are identical. See Figure 2). The objective function requires the number of distinct rows of M', or equivalently the cardinality of $\mathcal{H}$, to be minimum.

$$\mathcal{G} = \{(0221), (0011), (2102), (2220)\}$$

$$M = \left(\begin{array}{cccc} 0 & 2 & 2 & 1 \\ 0 & 0 & 1 & 1 \\ 2 & 1 & 0 & 2 \\ 2 & 2 & 2 & 0 \end{array}\right)\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} \qquad M' = \left(\begin{array}{cccc} 0 & \mathbf{1} & \mathbf{0} & 1 \\ 0 & \mathbf{0} & \mathbf{1} & 1 \\ \hline 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ \hline \mathbf{0} & 1 & 0 & \mathbf{1} \\ \mathbf{1} & 1 & 0 & \mathbf{0} \\ \hline \mathbf{1} & \mathbf{1} & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 \end{array}\right)\begin{array}{c} 1' \\ 1'' \\ 2' \\ 2'' \\ 3' \\ 3'' \\ 4' \\ 4'' \end{array}$$

$$\mathcal{H} = \{(0101), (0011), (1100), (0010)\}$$

Figure 2 Input ($\mathcal{G}$ and M) and Output (M' and $\mathcal{H}$) for PPH Problem

Problem 1 (*Pure Parsimony Haplotyping* (PPH)). Given a family $\mathcal{G}$ of genotypes, find a minimum-cardinality family $\mathcal{H}$ that explains $\mathcal{G}$.

In the following, let OPT denote the minimum size of an $\mathcal{H}$ that resolves $\mathcal{G}$.

Fact 2 (*First upper bound*. OPT $\le 2m$). Given any input family $\mathcal{G}$, there always exists an $\mathcal{H}$ that explains $\mathcal{G}$ with $|\mathcal{H}| \le 2|\mathcal{G}|$.

Proof. For a genotype $g \in \mathcal{G}$, let h_1 be the haplotype that is 0 whenever g is 0 and 1 in all other positions. Let h_0 be the haplotype that is 1 whenever g is 1 and 0 in all other positions. Then $g = h_0 \oplus h_1$. □

A family $\mathcal{H}'$ is said to be *a minimal family that explains* $\mathcal{G}$ if $\mathcal{H}'$ explains $\mathcal{G}$ and there exists no $\mathcal{H}'' \subset \mathcal{H}'$ that explains $\mathcal{G}$. We now show the simple result that a greedy algorithm for this problem computes a minimal (not necessarily minimum) solution. In fact, a greedy algorithm is a $\sqrt{m}$-approximation algorithm.

Fact 3 (*First lower bound*. OPT $> \sqrt{2m}$). Assume that $\mathcal{H}$ explains $\mathcal{G}$ and $|\mathcal{G}| \ge 2$. Then $|\mathcal{H}| > \sqrt{2m}$.

Proof. We show that $m \le (|\mathcal{H}|(|\mathcal{H}|-1))/2$. Indeed, if $\mathcal{H}$ explains $\mathcal{G}$, then to every g in $\mathcal{G}$ we can associate a different (unordered) pair of haplotypes in $\mathcal{H}$. Hence, $|\mathcal{G}| \le \binom{|\mathcal{H}|}{2} = (|\mathcal{H}|(|\mathcal{H}|-1))/2$. □

From any algorithm that computes a feasible solution it is possible to remove elements from the solution until a minimal solution is found. Any such technique guarantees a $\sqrt{m}$-approximation. For instance, the proof of Fact 2 suggests an algorithm:

Corollary 4. *Combining the proofs of Fact* 2 *and Fact* 3 *we get a greedy $\sqrt{m}$-approximation algorithm.*

Although it is easy to achieve a $\sqrt{m}$-approximation, it is difficult to obtain a δ-approximation for a constant δ. In fact, in the following section, we show that this task is impossible when δ is close to 1. We show that the PPH problem is APX-hard, and so there is a constant value $\bar{\delta}$ below which it is impossible to approximate the problem in polytime, unless P = NP (Ausiello et al. 1999). Determining the value $\bar{\delta}$ (or just a constant-approximation algorithm for PPH), is an interesting open problem. In this paper, we describe constant-approximation algorithms, but under the assumption that the number of occurrences of "2" in each genotype is bounded by a constant.

3. Complexity of the Problem

In this section we prove that the PPH problem is APX-hard. In particular, our proof shows that the problem is APX-hard even if each genotype has at most three ambiguous positions.

The reduction is from NODE-COVER, where we are given as input an undirected graph $G = (V, E)$ and we are asked to find a node cover $X \subseteq V$ of smallest possible cardinality. NODE-COVER is known to be APX-hard (Alimonti and Kann 1997, Berman and Karpinski 1998, Papadimitriou and Yannakakis 1991), even when the input instances are restricted to be graphs of degree at most three. Our arguments involve a classical theorem on node covers by Nemhauser and Trotter (1975).

3.1. NODE-COVER: The Theorem of Nemhauser and Trotter

Let $G = (V, E)$ be an undirected simple graph on $\hat{n}$ nodes and $\hat{m}$ edges. A *node cover* is a vertex-set $X \subseteq V$ such that every edge in E has at least one endpoint in X. Denote by σ_G the minimum size of a node cover in G.

As usual, if $S \subset V$, then $E[S] := \{uv \in E: u, v \in S\}$ denotes the set of edges with both endpoints in S, and $G[S]$ denotes *the subgraph of G induced by S*, i.e., $G[S] = (S, E[S])$. Moreover, when X is a node cover of G, then we say that X *covers* G. Below is a classical theorem on node covers which, although described in similar forms by several authors (see, e.g., Bar-Yehuda and Even 1985 or Chen et al. 2001), is mainly due to the original work by Nemhauser and Trotter (1975):

Theorem 5 (Nemhauser and Trotter 1975). *Given a graph $G = (V, E)$, introduce a new node v' for every node $v \in V$. Let $V' = \{v' : v \in V\}$ and $F = \{uv' : uv \in E\}$. Consider the bipartite graph $H = (V, V'; F)$ on $2|V|$ nodes and $2|E|$ edges. Let X be a minimum-cardinality node cover of H. Let $Y = \{v : v \in X$ and $v' \in X\}$ and $Z = \{v : v \in X$ or $v' \in X\}$. Then the following properties hold*:

(i) *if a set $D \subseteq Z$ covers $G[Z]$ then $D \cup Y$ covers G*;
(ii) *there exists a minimum cover of G that contains Y*;
(iii) $\sigma_{G[Z]} \ge |Z|/2$.

Define NC2 as the NODE-COVER problem restricted to graphs on which the optimal solution has size at least $\hat{n}/2$.

From Theorem 5, the following lemma holds.

Lemma 6. *The problem NC2 is APX-hard.*

Proof. We reduce the APX-hard problem NODE-COVER to NC2. Given an instance $G=(V,E)$ of NODE-COVER, the following polynomial algorithm (Procedure 1) defines an instance $G'=(V',E')$ of NC2 and a set $Q\subseteq V$ such that if X is a minimum node cover of G', then $X\cup Q$ is a minimum node cover of G:

Procedure 1 NC2reduction(G)

```
 0. $G' \leftarrow G$, i.e., $(V',E') \leftarrow (V,E)$;
 1. $Q \leftarrow \varnothing$;
 2. while (true)
 3.    compute from $G'$ the bipartite graph $H$
          and sets $Y$ and $Z$;
 4.    if ($Y \neq \varnothing$)
 5.       $G' \leftarrow (V'-Y, E'[V'-Y])$; $Q \leftarrow Q\cup Y$;
 6.    else if ($Z \neq V'$)
 7.       $G' \leftarrow (Z, E'[Z])$;
 8.    else
 9.       return $G'$ and $Q$;
10.    endif;
11. endwhile.
```

The loop 2–11 is executed at most $|V|$ times, and, because a minimum-cardinality node cover on a bipartite graph can be found in polynomial time, Step 3 is polynomial, so that the overall algorithm is polynomial. Step 5 is justified by property (ii) in Theorem 5. Similarly, in Step 7 we have that Y is empty, and $Z\subset V'$, so that we can apply property (i) of Theorem 5. When we reach Step 9, $Y=\varnothing$ and $Z=V'$ so that property (iii) of Theorem 5 holds, and we can exit the loop.

For an example of how Procedure 1 works, consider the graph G depicted in Figure 3. Start with $X=\{v_2', v_2, v_3', v_4'\}$ so that $Y=\{v_2\}$ and $Z=\{v_3, v_4\}$. This makes $Q=\{v_2\}$ and changes G so that v_2 is removed (and the only edge remaining is v_3v_4). In the next iteration, let $X=\{v_3, v_4\}$, so that $Y=\varnothing$ and $Z=\{v_3, v_4\}$. Then Q remains unchanged, and G reduces to the vertices v_3 and v_4, with the single edge v_3v_4. This graph is of class NC2 and is returned in the next iteration.

As for the approximability of NC2, assume that there exists an algorithm with performance guarantee λ for NC2. Then, from an approximate solution K for G' such that $|K|\le\lambda\sigma_{G'}$ we have that $|K\cup Q|\le \lambda\sigma_{G'}+|Q|\le\lambda(\sigma_{G'}+|Q|)=\lambda\sigma_G$. Hence, because NODE-COVER cannot be approximated arbitrarily close to 1 in polynomial time, neither can NC2, or we have a contradiction. Therefore, NC2 is APX-hard. □

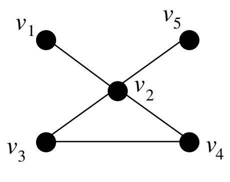

Figure 3 A Graph G

3.2. The APX-Hardness Proof for PPH

We now proceed to prove that PPH is APX-hard by reducing the problem NC2 to PPH. Given a graph $G=(V,E)$ that is an instance of NC2, we construct a matrix M with $\hat m+\hat n$ rows (genotypes) and $\hat n+1$ columns, where $\hat n=|V|$ and $\hat m=|E|$. We denote by $\mathcal{G}_M$ the set of rows (genotypes) of M. The rows are indexed by the elements in $V\cup E$ and the columns are indexed by the elements in $V\cup\{s\}$, where $s\notin V$ is a special symbol. The entries of M are defined by the following equations.

$$\begin{aligned}
M[u,u]&=0 && \text{for every } u\in V\\
M[u,v]&=1 && \text{for every } u,v\in V \text{ with } u\neq v\\
M[u,s]&=0 && \text{for every } u\in V\\
M[e,v]&=2 && \text{for every } v\in V \text{ and } e \text{ incident with } v\\
M[e,v]&=1 && \text{for every } v\in V \text{ and}\\
& && e\in E \text{ not incident at } v\\
M[e,s]&=2 && \text{for every } e\in E.
\end{aligned}$$

Lemma 7. *Let X be a node cover of G. Then there exists an $\mathcal{H}_X$ explaining $\mathcal{G}_M$ with $|\mathcal{H}_X|=\hat n+|X|$.*

Proof. For every $v\in V$, denote by h_v the haplotype that is 1 everywhere except at SNP v, where $h_v[v]=0$. Let X be a node cover of G. Consider the family $\mathcal{H}_X$ that contains as haplotypes the $\hat n$ rows of M indexed by elements in V and all the $|X|$ haplotypes of the form h_v, $v\in X$. Note that $\mathcal{H}_X$ explains $\mathcal{G}_M$. □

Lemma 8. *Let $\mathcal{H}$ be a minimum haplotype family explaining $\mathcal{G}_M$ such that the number of 0s in haplotypes in $\mathcal{H}$ is as small as possible. Then G admits a node cover $X_{\mathcal{H}}$ with $|X_{\mathcal{H}}|=|\mathcal{H}|-\hat n$.*

Proof. For every $v\in V$, denote by h_v the haplotype that is 1 everywhere except in column v, where $h_v[v]=0$. Let $\mathcal{H}$ be an optimal haplotype family explaining $\mathcal{G}_M$. Then $\mathcal{H}$ contains as haplotypes all the $\hat n$ rows of M indexed by elements in V because these genotypes are homozygous in each position. At this point, we observe that any row of M is either indexed by V (and hence is already explained) or by E (and hence is explained by introducing a single haplotype of the form h_v, where $v\in E$). Therefore, the lemma follows once we have proved the following claim.

Claim. Let h be a haplotype in $\mathcal{H}$ that is compatible with more than one row of M indexed by E. Then h is of the form h_v for some $v\in V$.

Proof. Clearly, h contains at most three 0s, or h would not be compatible with any row of M. Actually, h contains at most two 0s, because otherwise h would be compatible with at most one row of M indexed by E (we assume that G is simple). The same argument shows that if h has precisely two 0s, then one 0 is in the column indexed by s and that the other 0 is in the column indexed by node v. But then all rows of M indexed by E and compatible with h would correspond to edges incident at v and would hence all be explained by h_v plus the genotypes indexed by V, which are in $\mathcal{H}$. Hence, $\mathcal{H}\backslash\{h\} \cup \{h_v\}$ would also explain $\mathcal{G}_M$ and would contradict the minimality assumptions for $\mathcal{H}$.

Assume h has no component set to 0, that is, $h = \mathbf{1}$. Note however that haplotype $\mathbf{1}$ would only combine with haplotypes h' that have precisely three 0s, two of which correspond to the endnodes of some edge $e_{h'}$ in E. Moreover, as mentioned above, h' could not explain any genotype other than the one indexed by e. We can then remove h from $\mathcal{H}$ if we also replace each such h' in $\mathcal{H}$ with a haplotype h_v, with v being an endpoint of $e_{h'}$. This contradicts the optimality of $\mathcal{H}$.

Finally, assume that h has precisely one component set to 0. But then this component is not s, because otherwise this haplotype would combine only with haplotypes h' that have precisely two 0s corresponding to the endnodes of some edge $e_{h'}$ in E. Moreover h' could not explain any genotype other than the one indexed by e. We can then remove h from $\mathcal{H}$ by replacing each such h' in $\mathcal{H}$ with a haplotype h_v, with v being an endpoint of $e_{h'}$. This contradicts the optimality of $\mathcal{H}$. □

We complete the proof that PPH is APX-hard by showing that from an approximation algorithm with arbitrarily good performance guarantee for PPH follows an approximation algorithm with arbitrarily good performance guarantee for NC2, a contradiction.

Lemma 9. *Assume we have a $(1+\epsilon)$-approximation algorithm for PPH that holds on the restricted instances involved in the reduction proposed above. Then one can develop a $(1+3\epsilon)$-approximation algorithm for NC2.*

Proof. Assuming the minimum-cardinality node cover has size Opt, then there exists an $\mathcal{H}_{\text{opt}}$ explaining M with $|\mathcal{H}_{\text{opt}}| = \hat{n} + \text{Opt}$. By running the $(1+\epsilon)$-approximation algorithm for PPH we are guaranteed to find a solution $\mathcal{H}'$ with $|\mathcal{H}'| \le (\hat{n}+\text{Opt})(1+\epsilon)$. From this $\mathcal{H}'$ we have shown how one can find (in polynomial time) a node cover X of G of size at most

$$\begin{aligned}|X| &\le (\hat{n}+\text{Opt})(1+\epsilon) - \hat{n}\\ &= \epsilon\hat{n} + \text{Opt} + \epsilon\text{Opt}\\ &\le 2\epsilon\text{Opt} + \text{Opt} + \epsilon\text{Opt}\\ &= (1+3\epsilon)\text{Opt},\end{aligned}$$

where the third inequality follows from the fact that $\hat{n}/2 \le \text{Opt}$. □

4. Integer Linear-Programming Formulations

In this section, we describe two ILP formulations for the problem PPH. These formulations are used to derive exact algorithms, by means of some popular ILP solvers, such as CPLEX. The first formulation has an exponential number of variables and constraints, and is also described in Gusfield (2003), where it was shown that it can be used for practical instances. We review this formulation in §4.1. and we use it in §5 to obtain an approximation algorithm for PPH.

The second formulation is described in §4.2. and is the first model with a polynomial number of variables and constraints.

4.1. An Exponential-Size ILP Formulation

Denote by $\widehat{\mathcal{H}}_{\mathcal{G}}$ the set of haplotypes that are compatible with some genotype in $\mathcal{G}$. We associate a 0–1 variable x_h to every $h \in \widehat{\mathcal{H}}_{\mathcal{G}}$. The intended meaning of $x_h = 1$ is that h is in the solution, whereas $x_h = 0$ means h is not in the solution. Fix any total order on $\widehat{\mathcal{H}}_{\mathcal{G}}$ and denote by $\mathcal{P}$ the set of those pairs (h_1, h_2) with $h_1, h_2 \in \widehat{\mathcal{H}}_{\mathcal{G}}$, $h_1 < h_2$. Introduce a 0–1 variable y_{h_1,h_2} for every $(h_1, h_2) \in \mathcal{P}$. Moreover, for every $g \in \mathcal{G}$, let $\mathcal{P}_g := \{(h_1, h_2) \in \mathcal{P} \mid h_1 \oplus h_2 = g\}$.

The following is a valid ILP formulation of the PPH problem:

$$\text{OPT} = \min \sum_{h\in\widehat{\mathcal{H}}_{\mathcal{G}}} x_h \tag{1}$$

subject to

$$\sum_{(h_1,h_2)\in\mathcal{P}_g} y_{h_1,h_2} \ge 1 \quad \forall g \in \mathcal{G} \tag{2}$$

$$y_{h_1,h_2} \le x_{h_1} \quad \forall (h_1, h_2) \in \mathcal{P} \tag{3}$$

$$y_{h_1,h_2} \le x_{h_2} \quad \forall (h_1, h_2) \in \mathcal{P} \tag{4}$$

$$x \in \{0,1\}^{\widehat{\mathcal{H}}}, \quad y \in \{0,1\}^{\mathcal{P}}. \tag{5}$$

Because of constraints (2), each genotype is explained by at least one pair of haplotypes. Constraints (3) and (4) insure that (h_1, h_2) are allowed to explain g only if h_1 and h_2 are included in the solution.

4.2. A Polynomial-Size ILP Formulation

As mentioned in §2.1, the input of the problem is an $m \times n$ matrix M, with entries in $\{0, 1, 2\}$, while the output is a $2m \times n$ binary matrix M' with a minimum number of distinct rows. For each row g of M (genotype) denote by g' and g'' the two rows of M' (haplotypes) derived from the expansion of g (and hence such that $g = g' \oplus g''$). The order of the rows of M and M' induces a total ordering of the input genotypes and output haplotypes, which we will denote by $<$ in the following formulation.

We now describe an ILP formulation of PPH whose main decision variables correspond to each ambiguous position i of each genotype g. The setting of such variables identifies the values to give to $g'[i]$ and $g''[i]$ (one of them must be 0, and the other one must be 1). Let $\hat{\mathcal{G}} = \{g' \mid g \in \mathcal{G}\} \cup \{g'' \mid g \in \mathcal{G}\}$ be the set of rows of M'. Furthermore, for each ambiguous $g \in \mathcal{G}$ we define $\text{amb}(g) = \text{amb}(g') = \text{amb}(g'') := \{i \mid g[i] = 2\}$, the set of ambiguous positions of g. For each ambiguous $g \in \mathcal{G}$ and position $i \in \text{amb}(g)$ we define two 0–1 variables, $x^i_{g'}, x^i_{g''}$.

The problem asks us to set $x^i_{g'}$ to either 0 or 1, and $x^i_{g''}$ to $1 - x^i_{g'}$, i.e., there are constraints

$$x^i_{g'} + x^i_{g''} = 1 \quad \forall g \in G,\ i \in \text{amb}(g). \tag{6}$$

Now we need variables and constraints to count the distinct rows in M'. For simplicity, we first describe the model with more variables than is really needed.

Let $\mathcal{P}'$ be the set of pairs (a, b) such that $a, b \in \hat{\mathcal{G}}$ and $a < b$. For each pair $(a, b) \in \mathcal{P}'$ and each position i, we introduce a 0–1 variable

$$y^i_{ab} = \begin{cases} 1 & \text{if } a[i] = b[i] \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, for each $(a, b) \in \mathcal{P}'$, we introduce a 0–1 variable

$$y_{ab} = \begin{cases} 1 & \text{if rows } a \text{ and } b \text{ are identical} \\ 0 & \text{otherwise.} \end{cases}$$

Finally, for each row $r \in \hat{\mathcal{G}}$, we introduce a 0–1 variable

$$z_r = \begin{cases} 1 & \text{if there is no row } r' > r \text{ identical to } r \text{ in } M' \\ 0 & \text{otherwise.} \end{cases}$$

The objective function is then

$$\min \sum_{r \in \hat{\mathcal{G}}} z_r. \tag{7}$$

To describe the constraints, we first need a general result, concerning the product of two 0–1 variables. Given two binary variables u and v, we denote by $P(u, v)$ a binary variable such that $P(u, v) = uv$. This relation between $P(u, v)$, u, and v, can be obtained by forcing the following set of linear constraints, that we call $C(u, v)$:

$$[C(u,v){:}]\quad P(u,v) \le u; \quad P(u,v) \le v; \quad P(u,v) \ge u + v - 1. \tag{8}$$

We have the following constraints for y^i_{ab}:

$$y^i_{ab} = 1 \quad \forall (a,b) \in \mathcal{P}',\ i \in \{1, \ldots, n\} \mid i \notin (\text{amb}(a) \cup \text{amb}(b)) \wedge a[i] = b[i] \tag{9}$$

$$y^i_{ab} = 0 \quad \forall (a,b) \in \mathcal{P}',\ i \in \{1, \ldots, n\} \mid i \notin (\text{amb}(a) \cup \text{amb}(b)) \wedge a[i] \neq b[i] \tag{10}$$

$$y^i_{ab} = a[i]x^i_b + (1 - a[i])(1 - x^i_b) \quad \forall (a,b) \in \mathcal{P}',\ i \in \text{amb}(b),\ i \notin \text{amb}(a) \tag{11}$$

$$y^i_{ab} = b[i]x^i_a + (1 - b[i])(1 - x^i_a) \quad \forall (a,b) \in \mathcal{P}',\ i \in \text{amb}(a),\ i \notin \text{amb}(b) \tag{12}$$

$$y^i_{ab} = 2P(x^i_a, x^i_b) - x^i_a - x^i_b + 1 \quad \forall (a,b) \in \mathcal{P}',\ i \in \text{amb}(a) \cup \text{amb}(b). \tag{13}$$

These constraints are explained as follows. When i is not an ambiguous position for either a or b, y^i_{ab} is a constant, as shown in constraints (9) and (10). If i is ambiguous for only one of a and b, then, assuming $a[i]$ is ambiguous, we have that y^i_{ab} must be 1 if $x^i_a = b[i]$ and 0 otherwise. This is forced by constraints (11) and (12). Finally, if i is ambiguous for both a and b then y^i_{ab} must be 1 if $x^i_a = x^i_b$ and 0 otherwise. This is forced by constraints (13), together with the constraints $C(x^i_a, x^i_b)$ needed to impose $P(x^i_a, x^i_b) = x^i_a x^i_b$.

To complete the model, we need a set of constraints that ensure the intended meaning of y_{ab} and z_r. These are:

$$y_{ab} \le y^i_{ab} \quad \forall (a,b) \in \mathcal{P}',\ i = \{1, \ldots, n\} \tag{14}$$

$$y_{ab} \ge \sum_{i=1}^{n} y^i_{ab} - (n-1) \quad \forall (a,b) \in \mathcal{P}' \tag{15}$$

$$z_r \ge 1 - \sum_{l > r,\, l \in \hat{G}} y_{rl} \quad \forall r \in \hat{G}. \tag{16}$$

Because of constraints (14), if a and b disagree in any position, then they cannot be identical rows of M'. If, however, they are identical in all n positions, then they are identical rows, and this is forced by constraints (15). Finally, if $y_{rl} = 0$ for all $l > r$ it means that row $l > r$ has no identical row following it in M', and hence z_r must be 1. Note that we do not need to bound z_r from above because of the objective function.

In conclusion, the model for PPH is

$$\min \sum_{r \in \hat{\mathcal{G}}} z_r$$

subject to (6), (9)–(16) and to

$$C(x^i_a, x^i_b) \quad \forall a, b \in \hat{G},\ i \in \text{amb}(a) \cup \text{amb}(b) \tag{17}$$

with variables

$$x_a^i, y_{ab}^j, y_{ab}, z_a \in \{0,1\} \quad \forall (a,b) \in \mathcal{P}',\ i \in \operatorname{amb}(a),\ j \in \{1,\dots,n\}. \quad (18)$$

It is easy to see that there is a polynomial number of variables. In particular, there are $O(mn)$ variables x_a^i and $P(x_a^i, x_b^i)$, $O(m^2n)$ variables y_{ab}^i, $O(m^2)$ variables y_{ab}, and $O(m)$ variables z_a. As for the constraints, there are $O(mn)$ constraints (6) and (17), $O(m^2n)$ constraints (9)–(14), $O(m^2)$ constraints (15), and $O(m)$ constraints (16).

Overall, the model has $O(m^2n)$ variables and $O(m^2n)$ constraints. It is easy to reduce the number of both, by just observing that the decision variables are needed only for compatible haplotypes and ambiguous positions (otherwise, the variable can be replaced by a constant). In particular, let K be the total number of ambiguous positions in the input. Furthermore, let $\mathcal{R}$ be the set of pairs of compatible haplotypes in M' (a pair (a,b) is in $\mathcal{R}$ if a and b agree in all nonambiguous positions, and if they are not both the expansion rows of the same, ambiguous, genotype g). Pairs in $\mathcal{R}$ represent rows of M' that are possibly identical in the solution. Then, we only need to define variables y_{ab} and y_{ab}^i for $(a,b) \in \mathcal{R}$ and i that is an ambiguous position for a or b. Note that in M' there are overall $O(K)$ ambiguous positions. We obtain $O(Km)$ variables of type y_{ab}^i (for each ambiguous position $r[j]$ of M', there are at most $O(m)$ rows compatible with r), $O(|\mathcal{R}|)$ variables of type y_{ab}, and $O(K)$ variables of type x_a^i. Simple computations show that we obtain a model with a total of $O(Km)$ variables and constraints. Note that on average, R is smaller than m^2 and K is a fraction of mn (at most 1/2). K depends on the allele frequencies. For instance, if at a given SNP one allele has probability 3/4 and the other 1/4, a fraction of 6/16 individuals are heterozygous at that SNP on average.

5. An LP-Rounding 2^{k-1}-Approximation Algorithm

In this section we derive an approximation algorithm for PPH based on the ILP formulation described in §4.1.

Assume that every genotype $g \in \mathcal{G}$ contains at most k symbols of "2." Then the number of variables and inequalities in the ILP (1)–(5) is polynomial in the size of the input instance and we can solve the LP relaxation to optimality in polynomial time. Let z_{LP} be the optimal value of the LP relaxation, and (x^*, y^*) be an optimal LP solution. Perform the following two actions, in their sequential order, on the solution (x^*, y^*):

1. Scale the value of every variable by a factor of 2^{k-1};

2. If the value of a variable is at least 1, then round it to 1; otherwise, set the value of that variable to 0.

Let $(\bar{x}, \bar{y})$ denote the integer solution obtained by the above rounding procedure. We now argue that $(\bar{x}, \bar{y})$ is a feasible solution for PPH. In fact, because for any genotype g it is $|\mathcal{P}_g| \le 2^{k-1}$, then for at least one pair $(h_1, h_2) \in \mathcal{P}_g$ we have that $y^*_{(h_1,h_2)} \ge 1/2^{k-1}$. Therefore, $\bar{y}_{(h_1,h_2)} = 1$ for at least one pair $(h_1, h_2) \in \mathcal{P}_g$. Furthermore, whenever $\bar{y}_{(h_1,h_2)} = 1$, from constraints (3)–(4) it follows that $x^*_{h_1} \ge 2^{k-1}$ and $x^*_{h_2} \ge 2^{k-1}$. Then, $\bar{x}^*_{h_1} = 1$ and $\bar{x}^*_{h_2} = 1$, so that constraints (3)–(4) are satisfied by $(\bar{x}, \bar{y})$.

Because for each h it is $\bar{x}_h \le 2^{k-1}x_h^*$, the value of the objective function increases by at most a factor of 2^{k-1}. As a result of this deterministic rounding algorithm, we obtain a feasible solution of value at most $2^{k-1}z_{\mathrm{LP}} \le 2^{k-1}\mathrm{OPT}$, i.e., a 2^{k-1}-approximation.

6. A Randomized Approximation Algorithm

In this section we propose a randomized approximation algorithm for PPH, under the assumption that each genotype has at most k ambiguous sites. The algorithm has a slightly worse approximation ratio than the one based on linear programming described in §5. On the other hand, there are reasons to prefer approach of this section in some cases. In particular, the approach described here is simple to implement and, most of all, it does not require a third-party linear-programming solver.

We employ the mathematical symbol log for base 2 logarithms and reserve the mathematical symbol ln for logarithms on the natural base e. Recall that OPT denotes the optimal solution value, $m = |\mathcal{G}|$, and n is the number of SNPs. Let k be the maximum number of "2" characters in any $g \in \mathcal{G}$. The main result of this section is the following.

Theorem 10 (Monte Carlo). *There is a randomized algorithm that returns almost positively (i.e., with probability at least $(m-1)/m$) a feasible solution $\mathcal{H}_{feas}$ with*

$$|\mathcal{H}_{feas}| \le 2^{k+1}\lfloor \log m \rfloor (1 + \lceil \ln m \rceil)\mathrm{OPT}.$$

The running time of the algorithm is bounded by a polynomial in the size of the input instance. More precisely, Algorithm 4 runs in $O(m \log^3 m(n + \log m))$ time. (Hence it is an almost-linear-time algorithm.)

Informally, the proof of Theorem 10 is based on the following idea: If the solution has "low" cardinality, i.e., if OPT is "small" compared to m, then most genotypes in $\mathcal{G}$ are explained by haplotypes that are used to explain other genotypes in $\mathcal{G}$. Then, by randomly picking a suitable number of genotypes from $\mathcal{G}$ and computing all haplotypes compatible with them, we have a good chance of obtaining haplotypes that explain "many" of the genotypes in $\mathcal{G}$. We then remove these genotypes from $\mathcal{G}$. The idea is developed and explained in detail in §6.1., but there is a subtle obstacle that seems to prevent us from putting this simple idea into action: The suitable number of genotypes selected at random is a function of m, which we know, and also of OPT, which we do not know! To overcome this difficulty, we first obtain a weaker result in which we assume the existence of an oracle function, ORACLEOPT. The oracle, given a genotype family $\mathcal{G}$, returns the value of OPT for $\mathcal{G}$, denoted by OPT($\mathcal{G}$). So, in order to prove Theorem 10, the following partial result is first derived in §6.1.

THEOREM 11 (LAS VEGAS WITH ORACLE). *Assume* ORACLEOPT($\mathcal{G}$) *is an oracle that, given an instance $\mathcal{G}$ of PPH, returns the value* $\text{OPT} = \text{OPT}(\mathcal{G})$ *specifying the minimum number of haplotypes needed to explain $\mathcal{G}$. Then, there is a randomized algorithm that returns a feasible solution $\mathcal{H}_{feas}$ with* $|\mathcal{H}_{feas}| \le 2^{k+1}\lfloor \log m \rfloor (1 + \lceil \ln m \rceil)\text{OPT}$. *The running time of the algorithm is a concentrated random variable with expected value bounded by a polynomial in the size of the input instance.*

Because the average number of genotypes that are compatible with the same haplotype depends on both m and OPT, in the proof of Theorem 11, it is assumed that the value OPT is known. To obtain a true approximation algorithm, we must remove the dependence of the result from the knowledge of the actual value of OPT. How this is done is explained in §6.2., where the Monte Carlo algorithm of Theorem 10 is introduced and analyzed.

6.1. The Las Vegas Algorithm with Oracle

We now proceed to prove Theorem 11. Consider the following recursive algorithm (Algorithm 2).

Algorithm 2 HAPLOTYPE($\mathcal{G}$)

```
 1. if $\mathcal{G} = \varnothing$, return $\varnothing$;
 2. OPT ← ORACLEOPT($\mathcal{G}$);
 3. repeat
 4.    TEMP ← $\varnothing$;
 5.    for $2(1 + \lceil \ln m \rceil)$OPT times do
 6.       choose uniformly at random a $g \in \mathcal{G}$;
 7.       create, and put in TEMP, all haplotypes
             compatible with $g$;
 8.    endfor;
 9. until TEMP explains at least 1/2 of the
       genotypes in $\mathcal{G}$;
10. $\mathcal{G}' \leftarrow \{g \in \mathcal{G} : g$ is not explained by TEMP$\}$;
11. return TEMP ∪ HAPLOTYPE($\mathcal{G}'$).
```

In the following lemma we settle the correctness and approximation-guarantee issues.

LEMMA 12. *When Algorithm 2 terminates, then it returns a haplotype family $\mathcal{H}_{\text{feas}}$ that explains $\mathcal{G}$. Moreover,* $|\mathcal{H}_{\text{feas}}| \le 2^{k+1}\lfloor \log m \rfloor (1 + \lceil \ln m \rceil)\text{OPT}$.

PROOF. By induction on $m = |\mathcal{G}|$. If $m = 0$, then HAPLOTYPE($\mathcal{G}$) halts at Step 1 and returns an optimal solution of size 0, which provides the base for the induction. For the inductive step, assume that $m > 0$ and that the claim in the lemma holds for $m' < m$. The claim also assumes that HAPLOTYPE($\mathcal{G}$) halts, which for $m > 0$, necessarily occurs at Step 11. Hence, $|\mathcal{G}'| \le 1/2|\mathcal{G}| < |\mathcal{G}|$ by the test at Step 9. Therefore, we can assume by induction that HAPLOTYPE($\mathcal{G}'$) returns a set of haplotypes that explains $\mathcal{G}'$. Because TEMP explains all genotypes in $\mathcal{G} \backslash \mathcal{G}'$, then HAPLOTYPE($\mathcal{G}$) returns a haplotype family that explains all genotypes in $\mathcal{G}$. This happens at Step 11, where TEMP ∪ HAPLOTYPE($\mathcal{G}'$) is returned. To prove the approximation guarantee, we have to bound the size of TEMP ∪ HAPLOTYPE($\mathcal{G}'$). By induction, the size of HAPLOTYPE($\mathcal{G}'$) is bounded by $2^{k+1}\lfloor \log \lfloor m/2 \rfloor \rfloor (1 + \lceil \ln m \rceil)\text{OPT}$, because $|\mathcal{G}'| \le (1/2)|\mathcal{G}| = m/2$ and any haplotype family that explains $\mathcal{G}$ also explains its subset $\mathcal{G}'$. To bound the size of TEMP, remember that each genotype g in $\mathcal{G}$ has at most k ambiguous sites, hence there are at most 2^k haplotypes compatible with g. Because TEMP is each time reset to an empty set at Step 4, and iterations 5–8 choose at most $2(1 + \lceil \ln m \rceil)\text{OPT}$ different genotypes from $\mathcal{G}$, the number of haplotypes in TEMP never exceeds $2^k 2(1 + \lceil \ln m \rceil)\text{OPT} = 2^{k+1}(1 + \lceil \ln m \rceil)\text{OPT}$. Therefore,

$$\begin{aligned}
|\mathcal{H}_{feas}| &= |\text{TEMP} \cup \text{HAPLOTYPE}(\mathcal{G}')| \\
&\le |\text{TEMP}| + |\text{HAPLOTYPE}(\mathcal{G}')| \\
&\le 2^{k+1}(1 + \lceil \ln m \rceil)\text{OPT} + 2^{k+1}\left\lfloor \log \left\lfloor \frac{m}{2} \right\rfloor \right\rfloor \\
&\quad \cdot (1 + \lceil \ln m \rceil)\text{OPT} \\
&\le 2^{k+1}\left\lfloor 1 + \log \left\lfloor \frac{m}{2} \right\rfloor \right\rfloor (1 + \lceil \ln m \rceil)\text{OPT} \\
&= 2^{k+1}\lfloor \log m \rfloor (1 + \lceil \ln m \rceil)\text{OPT}. \quad \square
\end{aligned}$$

We are now left with the analysis of the running time. This takes the remainder of this section and motivates the design of the Monte Carlo algorithm introduced in §6.2.

Denote by $\mathcal{H}_{\text{opt}}$ an optimal solution to the haplotyping problem, i.e., $\text{OPT} = |\mathcal{H}_{\text{opt}}|$. Denote by $\widehat{\mathcal{H}}_{\mathcal{G}}$ the set of those haplotypes that are compatible with some genotype in $\mathcal{G}$. Clearly, $\mathcal{H}_{\text{opt}} \subseteq \widehat{\mathcal{H}}_{\mathcal{G}}$.

To understand the following analysis better, it is useful to think of a graph on node set $\widehat{\mathcal{H}}_{\mathcal{G}}$ and having an edge h_1h_2 labeled with $g \in \mathcal{G}$ whenever $g = h_1 \oplus h_2$. (This graph is simple and a set of edges labeled with a same genotype forms a matching.) Denote by $d(h)$ the number of genotypes in $\mathcal{G}$ compatible with $h \in \widehat{\mathcal{H}}_{\mathcal{G}}$. We define a haplotype in $\mathcal{H}_{\text{opt}}$ to be *good* when the fraction of genotypes in $\mathcal{G}$ is compatible with h is at least $m/(2\text{OPT})$.

Definition 13. A haplotype h in $\mathcal{H}_{\text{opt}}$ is *good* when $d(h) \geq m/(2\text{OPT})$ and *bad* otherwise.

A first motivation to the above definition comes from the following lemma.

Lemma 14. *Let $\mathcal{H}_{\text{opt}}^{\text{good}}$ be the set of good haplotypes in $\mathcal{H}_{\text{opt}}$. Then at least half of the genotypes in $\mathcal{G}$ are explained by $\mathcal{H}_{\text{opt}}^{\text{good}}$.*

Proof. All the genotypes in $\mathcal{G}$ are explained by $\mathcal{H}_{\text{opt}}$. Hence, the genotypes in $\mathcal{G}$ that are not explained by $\mathcal{H}_{\text{opt}}^{\text{good}}$ are at most $\sum_{h\in\mathcal{H}_{\text{opt}}\setminus\mathcal{H}_{\text{opt}}^{\text{good}}} d(h)$.

Because $d(h) < m/(2\text{OPT})$ holds for every bad haplotype h, the genotypes in $\mathcal{G}$ that are not explained by $\mathcal{H}_{\text{opt}}^{\text{good}}$ are at most

$$\sum_{h\in\mathcal{H}_{\text{opt}}\setminus\mathcal{H}_{\text{opt}}^{\text{good}}} d(h) < \sum_{h\in\mathcal{H}_{\text{opt}}\setminus\mathcal{H}_{\text{opt}}^{\text{good}}} \frac{m}{2\text{OPT}}$$
$$= \left|\mathcal{H}_{\text{opt}}\setminus\mathcal{H}_{\text{opt}}^{\text{good}}\right| \frac{m}{2\text{OPT}} < \text{OPT}\frac{m}{2\text{OPT}} = \frac{m}{2} = \frac{|\mathcal{G}|}{2}. \quad \square$$

For a second motivation, consider the following observation.

Observation 15. When h is good, the fraction of genotypes in $\mathcal{G}$ that are compatible with h is at least $1/(2\text{OPT})$.

Proof. The fraction of those genotypes in $\mathcal{G}$ that are compatible with h is

$$\frac{d(h)}{|\mathcal{G}|} \geq \frac{m/(2\text{OPT})}{m} = \frac{1}{2\text{OPT}}. \quad \square$$

As a consequence, when we choose a random $g \in \mathcal{G}$ in Step 6, the probability that any fixed, good haplotype h is put in Temp at Step 7 is at least $1/(2\text{OPT})$. Because within loop 5–8 this experiment is repeated $2(1+\lceil \ln m\rceil)\text{OPT}$ times, we expect that all good haplotypes end up in Temp within a full execution of loop 5–8. This is more formally stated in the following lemma.

Lemma 16. *Each time we reach Step 9 we have*

$$P\left[\mathcal{H}_{\text{opt}}^{\text{good}} \subseteq \text{Temp}\right] \geq (e-2)/e.$$

Proof. Let h be a good haplotype. We first show that when we get to the test in Step 9, after running the $2(1+\lceil \ln m\rceil)\text{OPT}$ iterations of loop 5–8, it is $P[h \notin \text{Temp}] \leq 1/(em)$. From this it easily follows that the probability of any good haplotype in $\mathcal{H}_{\text{opt}}$ not being in Temp is at most $2/e$.

Because h is good, and the random choices are independent,

$$P[h\notin \text{Temp}] \leq \prod_{i=1}^{2(1+\lceil \ln m\rceil)\text{OPT}} \left(1-\frac{1}{2\text{OPT}}\right)$$
$$= \left(1-\frac{1}{2\text{OPT}}\right)^{2(1+\lceil \ln m\rceil)\text{OPT}}$$
$$= \left[\left(1-\frac{1}{2\text{OPT}}\right)^{2\text{OPT}}\right]^{1+\lceil \ln m\rceil}$$
$$\leq \left(\frac{1}{e}\right)^{1+\lceil \ln m\rceil} \leq \frac{1}{em}.$$

To bound the probability that any good haplotype does not make it into Temp, we employ the simple fact that the probability of a union of events is at most the sum of the probabilities of the events themselves, no matter what the dependencies are. Hence,

$$P\left[\exists h\in\mathcal{H}_{\text{opt}}^{\text{good}}\setminus\text{Temp}\right] \leq \sum_{h\in\mathcal{H}_{\text{opt}}^{\text{good}}} P[h\notin\text{Temp}]$$
$$\leq \sum_{h\in\mathcal{H}_{\text{opt}}^{\text{good}}} \frac{1}{em} = \frac{\left|\mathcal{H}_{\text{opt}}^{\text{good}}\right|}{em}$$
$$\leq \frac{|\mathcal{H}_{\text{opt}}|}{em} = \frac{\text{opt}}{em} \leq \frac{2}{e}. \quad \square$$

The last inequality in the above chain is due to Fact 2.

Corollary 17. *Each time we get to the test at Step 9, the probability of passing the test and exiting is at least $(e-2)/e$.*

Proof. By Lemma 16, each time the test at Step 9 is faced we have $P[\mathcal{H}_{\text{opt}}^{\text{good}} \subseteq \text{Temp}] \geq (e-2)/e$. By Lemma 14, this means that the probability that the test is positive is at least $(e-2)/e$. $\square$

From Corollary 17 it easily follows that the running time of Algorithm 2 is a concentrated random variable with expected value bounded by a polynomial in the size of the input instance. We omit the formal proof because these arguments do not affect the design and analysis of the Monte Carlo algorithm, which is most relevant in this section.

6.2. The Monte Carlo Algorithm

The main idea behind the Monte Carlo algorithm is to perform a binary search on OPT at each recursion of Algorithm 2. To do so, we introduce a procedure that tests a candidate value for OPT. We call this procedure TryOPTvalue. Procedure TryOPTvalue receives as input a genotype family $\mathcal{G}$, a hypothesis value OPT′ for OPT, and a parameter numAttempts. If procedure TryOPTvalue succeeds in finding a haplotype family that explains at least half of $\mathcal{G}$ under the hypothesis value OPT′ within at most numAttempts attempts, then the procedure returns this haplotype family. Otherwise the procedure returns a NO answer.

Procedure 3 TryOPTvalue ($\mathcal{G}$, OPT′, numAttempts)
```
1. for $t := 1$ to numAttempts do
2.    Temp $\leftarrow \varnothing$;
3.    for $2(1+\lceil \ln m\rceil)\text{OPT}'$ times do
4.       choose uniformly at random a $g \in \mathcal{G}$;
```

```
5.        create, and put into TEMP, all haplotypes
              compatible with $g$;
6.     endfor;
7.     if TEMP explains at least 1/2 of the
           genotypes in $\mathcal{G}$ then
8.        return TEMP;
9.  endfor;
10. return NO;
```

With respect to the above procedure, we prove the following lemma.

LEMMA 18. *Let $\mathcal{G}$ be any genotype family and* OPT' *be any integer with* $\mathrm{OPT}' \geq \mathrm{OPT}(\mathcal{G})$. *Then the probability that* TRYOPTVALUE $(\mathcal{G}, \mathrm{OPT}', e \ln m)$ *will return a NO answer is at most* $1/m^2$.

PROOF. From Corollary 17, each time the test at Step 7 is faced, the probability of success is at least $(e-2)/e$. Therefore, the probability that TRYOPTVALUE $(\mathcal{G}, \mathrm{OPT}', e \ln m)$ will return a NO answer is bounded by

$$\left(1-\frac{2}{e}\right)^{e\ln m} = \left(1-\frac{2}{e}\right)^{(e/2)2\ln m} \leq \left(\frac{1}{e}\right)^{2\ln m} = \frac{1}{m^2}. \quad \square$$

REMARK 19. Let $m = |\mathcal{G}|$ and n be the number of SNPs. Assume $\mathrm{OPT}' \leq 2m$. Then running TRYOPTVALUE $(\mathcal{G}, \mathrm{OPT}', T)$ takes $O(Tm \log m(n + \log m))$ time.

PROOF. The T factor is due to the number of iterations of loop 1–9 and the $m \log m$ factor is due to the number of iterations of loop 3–6. We assume that randomly choosing an object among m takes $O(\log m)$ time. Producing all haplotypes that are compatible with a given genotype g requires $O(2^k n) = O(n)$ time. $\square$

In the following, we denote by DOUBLE$(\mathcal{G})$ a procedure that, given as input a genotype family $\mathcal{G}$, returns a haplotype family $\mathcal{H}_{feas}$ that explains $\mathcal{G}$ with $|\mathcal{H}_{\text{feas}}| \leq 2|\mathcal{G}|$. (The existence of such a procedure is guaranteed by Fact 2.) We are now ready to propose an efficient Monte Carlo algorithm.

Algorithm 4 MCHAPLO$(\mathcal{G})$

```
1.  if $\mathcal{G} = \varnothing$ then return $\varnothing$;
2.  $\mathrm{OPT}_1 \leftarrow \sqrt{m}$; $\mathrm{OPT}_2 \leftarrow 2m$; $\mathcal{H}_{temp} \leftarrow$ DOUBLE$(\mathcal{G})$;
3.  while $\mathrm{OPT}_2 > \mathrm{OPT}_1 + 1$ do
4.     $\mathrm{OPT}' \leftarrow \lfloor (\mathrm{OPT}_1 + \mathrm{OPT}_2)/2 \rfloor$;
5.     TEMP $\leftarrow$ TRYOPTVALUE $(\mathcal{G}, \mathrm{OPT}', e \ln m)$;
6.     if TEMP is a NO answer then $\mathrm{OPT}_1 \leftarrow \mathrm{OPT}'$;
7.     else set $\mathrm{OPT}_2 \leftarrow \mathrm{OPT}'$ and $\mathcal{H}_{temp} \leftarrow$ TEMP;
8.  endwhile;
9.  $\mathcal{G}' \leftarrow \{g \in \mathcal{G} : g$ is not explained by $\mathcal{H}_{temp}\}$;
10. return $\mathcal{H}_{temp} \cup$ MCHAPLO$(\mathcal{G}')$.
```

REMARK 20. Algorithm 4 is an almost-linear-time algorithm. More precisely, Algorithm 4 runs in $O(m \log^3 m(n + \log m))$ time.

PROOF. By Remark 19, the time required for the first recursion is $O(m \log^3 m(n + \log m))$ because TRYOPTVALUE $(\mathcal{G}, \mathrm{OPT}', e \ln m)$ is called $O(\log m)$ times. This dominates the time for the whole execution because m halves at each recursion. $\square$

The next lemma closes our analysis of Algorithm 4.

LEMMA 21. *Let $\mathcal{G}$ be a genotype family. Then the haplotype family returned by* MCHAPLO$(\mathcal{G})$ *explains $\mathcal{G}$. Moreover, the probability that the size of the haplotype family* MCHAPLO$(\mathcal{G})$ *exceeds* $2^{k+1} \lfloor \log m \rfloor (1 + \lceil \ln m \rceil)\mathrm{OPT}$ *is at most* $1/m$.

PROOF. It is clear that MCHAPLO$(\mathcal{G})$ explains $\mathcal{G}$. Moreover, we know that $|\text{MCHAPLO}(\mathcal{G})| \leq 2^{k+1} \cdot \lfloor \log m \rfloor (1 + \lceil \ln m \rceil)\mathrm{OPT}$ holds if the execution of the algorithm never observes the following situation:

> There has been some call TRYOPTVALUE $(\mathcal{G}^*, \mathrm{OPT}', e \ln m^*)$ with $m^* = |\mathcal{G}^*|$ and $\mathrm{OPT}' \geq \mathrm{OPT}(\mathcal{G}^*)$ that returned a NO answer.

Note that, during the execution of MCHAPLO$(\mathcal{G})$, the total number of calls to Procedure TRYOPTVALUE is at most $\log m$. Moreover, by Lemma 18, for all those calls in which $\mathrm{OPT}' \geq \mathrm{OPT}(\mathcal{G}^*)$, the probability of obtaining a NO answer is at most $1/m^2$. Therefore,

$$P[|\text{MCHAPLO}(\mathcal{G})| > 2^{k+1} \lfloor \log m \rfloor (1 + \lceil \ln m \rceil)\mathrm{OPT}] \leq \log m \frac{1}{m^2} \leq \frac{1}{m}. \quad \square$$

7. Conclusions

In this paper we studied the pure parsimony haplotyping problem, with investigations into (i) complexity, (ii) exact algorithms, and (iii) approximation algorithms. There are directions for future research with respect to all these aspects.

(i) We have shown that the problem is APX-hard when each genotype has at most three ambiguous sites. What about the case of genotypes with at most two ambiguous sites? Is the problem then polynomially solvable?

(ii) The exact approach based on ILP has its limitations, mainly due to the high number of variables and constraints (also in the case of the polynomial-sized formulation), which makes it of little use for instances with a large number of ambiguous sites. It is possible that a combinatorial branch-and-bound algorithm may be preferable over ILP approaches for this problem. This issue deserves further investigation.

(iii) We developed approximation algorithms for the case of a bounded number, k, of ambiguous sites. Our approximation ratios depend exponentially on k. Is it possible to achieve ratios that depend polynomially on k? Also, can we remove at least part of the $2^2 \lfloor \log m \rfloor (1 + \lceil \ln m \rceil)$ overhead of the randomized algorithm with respect to the approximation algorithm based on linear programming? Finally, the most interesting question about approximation algorithms is: Can we design a constant δ-approximation algorithm for the PPH problem?

Acknowledgments
The authors thank the referees for a careful reading of this paper and many helpful suggestions. Giuseppe Lancia thanks the Department of Computer Science, University of Trento, for their hospitality during the period when most of this work was completed, and the Mathematical Biosciences Institute of Columbus, Ohio, for their hospitality during the period when this work was revised.

References

Alimonti, P., V. Kann. 2000. Hardness of approximating problems on cubic graphs. *Theroet. Comput. Sci.* **237** 123–134.

Ausiello, G., P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi. 1999. *Complexity and Approximation. Combinatorial Optimization Problems and Their Approximability Properties*. Springer, Berlin, Germany.

Bafna, V., D. Gusfield, G. Lancia, S. Yooseph. 2003. Haplotyping as perfect phylogeny: A direct approach. *J. Comput. Biol.* **10** 323–340.

Bar-Yehuda, R., S. Even. 1985. A local-ratio theorem for approximating the weighted vertex cover problem. *Ann. Discrete Math.* **25** 27–46.

Berman, P., M. Karpinski. 1998. On some tighter inapproximability results. Technical report ECCC No. 29, Department of Computer Science, University of Trier, Trier, Germany.

Chakravarti, A. 1998. It's raining SNP, hallelujah? *Nature Genetics* **19** 216–217.

Chen, J., I. A. Kanj, W. Jia. 2001. Vertex cover: Further observations and further improvements. *J. Algorithms* **41** 280–301.

Clark, A. 1990. Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular Biol. Evolution* **7** 111–122.

Eskin, E., E. Halperin, R. Karp. 2003. Efficient reconstruction of haplotype structure via perfect phylogeny. *J. Bioinformatics Comput. Biol.* **1** 1–20.

Gusfield, D. 2000. A practical algorithm for optimal inference of haplotypes from diploid populations. *Annual Internat. Conf. Intelligent Systems Molecular Biol.* AAAI Press, Menlo Park, CA, 183–189.

Gusfield, D. 2001. Inference of haplotypes from samples of diploid populations: Complexity and algorithms. *J. Comput. Biol.* **8** 305–324.

Gusfield, D. 2003. Haplotype inference by pure parsimony. *Annual Sympos. Combin. Pattern Matching*. *Springer Lecture Notes in Computer Science*, No. 2676. Springer-Verlag, Berlin, Germany, 144–155.

Helmuth, L. 2001. Genome research: Map of the human genome 3.0. *Science* **293** 583–585.

Hubbel, E. 2002. Personal communication.

International Human Genome Sequencing Consortium. 2001. Initial sequencing and analysis of the human genome. *Nature* **409** 860–921.

Lancia, G., V. Bafna, S. Istrail, R. Lippert, R. Schwartz. 2001. SNPs problems, complexity and algorithms. *Annual Eur. Sympos. Algorithms. Springer Lecture Notes in Computer Science*, No. 2161. Springer-Verlag, Berlin, Germany, 182–193.

Li, L., J. H. Kim, M. Waterman. 2003. Haplotype reconstruction from SNP alignment. *ACM Annual Internat. Conf. Comput. Molecular Biol.* ACM Press, New York, 207–216.

Lippert, R., R. Schwartz, G. Lancia, S. Istrail. 2002. Algorithmic strategies for the SNPs haplotype assembly problem. *Briefings Bioinformatics* **3** 23–31.

Marshall, E. 1999. Drug firms to create public database of genetic mutations. *Sci. Magazine* **284** 406–407.

Nemhauser, G. L., L. E. Trotter. 1975. Vertex packings: Structural properties and algorithms. *Math. Programming* **8** 232–248.

Papadimitriou, C. H., M. Yannakakis. 1991. Optimization, approximation, and complexity classes. *J. Comput. System Sci.* **43** 425–440.

Rizzi, R., V. Bafna, S. Istrail, G. Lancia. 2002. Practical algorithms and fixed-parameter tractability for the single individual SNP haplotyping problem. *Annual Workshop on Algorithms in Bioinformatics. Springer Lecture Notes in Computer Science*, No. 2452. Springer-Verlag, Berlin, Germany, 29–43.

Venter, J. C., et al. 2001. The sequence of the human genome. *Science* **291** 1304–1351.