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

Giuseppe Lancia[1] Cristina Maria Pinotti[2] Romeo Rizzi[2]

1. *Dipartimento di Matematica e Informatica, Università di Udine*
lancia@dimi.uniud.it
2. *Dipartimento di Informatica e Telecomunicazioni, Università di Trento*
{romeo,pinotti}@science.unitn.it

Abstract

In this paper we address the *Pure Parsimony Haplotyping* problem: Given a set of genotypes $\mathcal{G}$, find a minimum cardinality set of haplotypes which explains $\mathcal{G}$. A genotype g is an n-dimensional vector in $\{0,1,2\}$ and a haplotype h is an n-dimensional binary vector. A set of haplotypes $\mathcal{H}$ is said to explain $\mathcal{G}$ if for every $g \in \mathcal{G}$ there are $h_1, h_2 \in \mathcal{H}$ such that $h_1 \oplus h_2 = g$. The position-wise sum $h_1 \oplus h_2$ indicates the genotype which has a 2 in the positions where h_1 and h_2 disagree, and the same value as h_1 and h_2 where they agree. In this paper we show the APX-hardness of the problem even when the number of "2" symbols is at most 3 for every $g \in \mathcal{G}$. We then give two Integer Linear Programming formulations. From the first one (appeared also in [9]) we derive a 2^{k-1}-approximation algorithm when the number of symbols "2" is at most k for every $g \in \mathcal{G}$. This formulation has an exponential number of variables and constraints. The second formulation is new, and has a polynomial number of variables and constraints. Finally, we give approximation algorithms not based on Linear Programming, such as a trivial $\sqrt{|\mathcal{G}|}$-approximation algorithm for the general case, and an effective probabilistic algorithm with a performance guarantee of $2^{k+1}\lfloor \log |\mathcal{G}| \rfloor (1 + \lceil \ln |\mathcal{G}| \rceil)$ when the number of symbols "2" is at most k for every $g \in \mathcal{G}$. The expected running time of the algorithm turns out to be almost linear in the input size.

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 [21, 12] 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 [4].

Since 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. Nowadays, 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 [16, 10].

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 either be *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 in 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, the recent years have seen the birth of a set of new combinatorial and optimization problems related to SNPs. In particular, since 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* [13, 15, 20, 14]).

The cheapest way to haplotype a population (i.e., a set of individuals), is to first 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 hetero- zygous. The ambiguity comes from heterozygous sites, since, 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 to determine two haplotypes such that, if they are assumed to be the ones on the two chromosome copies, then, computing the multiplicity of each SNP allele we obtain exactly the genotype g. The *Population Haplotyping Problem* is then the following: 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, (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 [2, 6].

- *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*. In 1990, the biologist A. Clark suggested a greedy rule [5] 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 [7, 8].

- *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 [9], who adopted an Integer Programming formulation for its practical solution. The problem is NP-hard, as first shown by Hubbel [11].

Each model and objective function above 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* for a set of haplotypes 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 [9].

1.1 Our results and paper organization

The problem we study is the following: find a set $\mathcal{H}$ of haplotypes of *smallest* cardinality to explain a given set $\mathcal{G}$ of m genotypes. The binary nature of SNPs leads to a nice combinatorial problem. As described in the sequel, 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 (see also [11]), but, in fact, 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 Section 3.

Because of its complexity, any exact algorithm for the problem must resort to some sort of enumeration. In Section 4 we describe two Integer Linear Programming (ILP) formulations that can be adopted for the problem solution. The first formulation, described in Section 4.1, has an exponential number of variables and constraints, and was used in [9] to obtain a practical solution for problem instances of moderate (but still important for real-life applications) size. Later in the paper we will use this formulation to obtain an approximation algorithm based on Linear Programming. The second ILP formulation, described in Section 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 Section 4.1, in Section 5 we describe a 2^{k-1}-approximation algorithm, based on deterministic rounding of an optimal Linear Programming solution.

By using a very different approach we were able to obtain a similar result. The approach, based on random sampling, does not require the use of an LP-solver code, which is sometimes not available or too expensive to obtain. In Section 6.2 we describe a (Monte Carlo) randomized polynomial algorithm, which returns almost surely (i.e. with probability at least $\frac{m-1}{m}$) a feasible solution $\mathcal{H}_{feas}$ with $|\mathcal{H}_{feas}| \leq 2^{k+1}\lfloor \log m \rfloor (1 + \lceil \ln m \rceil)\,\mathrm{OPT}$. The running time of the algorithm is bounded by a polynomial in the size of the input instance. More precisely, 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 better introduce and motivate our Monte Carlo algorithm, we will first design another randomized (Las Vegas) algorithm, described in Section 6.1, which, assuming the ability to know the OPT value, constructs a feasible solution $\mathcal{H}_{feas}$ with $|\mathcal{H}_{feas}| \leq 2^{k+1}\lfloor \log m \rfloor (1 + \lceil \ln m \rceil)\,\mathrm{OPT}$. The running time of the algorithm is a random variable with expected value bounded by a polynomial in the size of the input instance.

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

Haplotype 1 $\rightarrow$	`C`	`C`	`A`
Haplotype 2 $\rightarrow$	`G`	`C`	`T`

Figure 1: A chromosome and the two haplotypes

2 Preliminaries: SNPs, haplotypes and genotypes

A Single Nucleotide Polymorphism, or SNP, is a position in the genome at which some of us have a certain base while the others have a different one. The two base values are called alleles. Due to the binary nature of SNPs, we can 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) which 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 a given encoding of the alleles, these two haplotypes could be represented, e.g., 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" at SNP 1 stands for `G`.

Haplotype data is particularly sought after in the study of complex diseases (those affected by more than a single gene), since it can give complete information about which set of gene alleles are inherited together. 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 is 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 of a symbol 0 (1) corresponds to an homozygous site for the allele 0 (respectively, 1), and each position of the symbol 2 (called an *ambiguous* position) corresponds to an heterozygous site. A *haplotype* is a *vector with entries in* $\{0, 1\}$ (i.e., a binary vector). In the sequel, n will denote the length of genotype and haplotype vectors.

2.1 Problem definition

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

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

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

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

Figure 2: Input ($\mathcal{G}$ and M) and output (M' and $\mathcal{H}$) for (PPH) problem.

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

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

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 (or *good* for) a genotype g if h agrees with g at all non-ambiguous positions.

The problem input is a *population*, that is, a family $\mathcal{G} = \{g_1, \dots, g_m\}$ of m genotypes, on n SNPs. The input can also be seen 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 will be useful in formalizing an Integer Programming model of the problem, in Section 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 which explains $\mathcal{G}$. Alternatively, the output can be seen as a $2m \times n$ binary matrix, in which each row is a haplotype and there is a 1-to-1 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 $g' \oplus g'' = g$. See Figure 2). The objective function requires the number of distinct rows of M', or, equivalently, the cardinality of $\mathcal{H}$, to be minimum:

Problem 1 (PURE PARSIMONY HAPLOTYPING (PPH)). *Given a family $\mathcal{G}$ of genotypes, find a minimum-cardinality family $\mathcal{H}$ which explains $\mathcal{G}$.*

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

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

Proof: For a genotype $g \in \mathcal{G}$, let h_1 be the haplotype which is 0 whenever g is 0 and 1 in all other positions. Let h_0 be the haplotype which 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 which explains* $\mathcal{G}$ if $\mathcal{H}'$ explains $\mathcal{G}$ and there exists no $\mathcal{H}'' \subset \mathcal{H}'$ that explains $\mathcal{G}$. Note that for any minimal family $\mathcal{H}'$, it is $|\mathcal{H}'| \leq 2m$. We now show a very simple result, i.e., that basically any greedy algorithm for this problem, that computes a minimal (not necessarily minimum) solution, is a $\sqrt{m}$-approximation algorithm.

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

Proof: We will show that $m \leq \frac{|\mathcal{H}|(|\mathcal{H}|-1)}{2}$. 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, $|\mathcal{G}| \leq \binom{|\mathcal{H}|}{2} = \frac{|\mathcal{H}|(|\mathcal{H}|-1)}{2}$. □

From any algorithm that computes a feasible solution it is possible, by removing elements from the solution until a minimal solution is left, to guarantee a $\sqrt{m}$-approximation. For instance, the proof of Fact 2 suggests an algorithm:

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

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

3 The complexity of the problem

In this section we prove that the problem (PPH) is APX-hard, i.e., there cannot be an approximation algorithm for this problem with a performance guarantee arbitrarily close to 1, unless P=NP [1]. 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 [19, 18, 3] to be APX-hard, 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 [17].

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, when $S \subset V$, then $G[S]$ denotes *the subgraph of G induced by S*, i.e. $G[S] = (S, \{uv \in E : u, v \in S\})$. Moreover, when X is a node cover of G, then we say that X *covers* G. Here follows a classical theorem on node covers by Nemhauser and Trotter [17].

Theorem 5 (Nemhauser and Trotter [17]). *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 $2\hat{n}$ node and $2\hat{m}$ edges. Let X be a minimum node cover of H. Let $Y = \{v : v \in X \wedge\ v' \in X\}$ and $Z = \{v : v \in X \vee\ v' \in X\}$. Then the following properties hold:*

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

□

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

From the above Theorem, 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:

The loop 2–11 is executed at most $|V|$ times, and, since a minimum 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., $Y = \emptyset$ and $Z = V'$ so that property (iii) of Theorem 5 holds, and we can exit the loop.

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 $|K| \leq \lambda\, \sigma_{G'}$ we have that $|K \cup Q| \leq \lambda\, \sigma_{G'} + |Q| \leq \lambda\, (\sigma_{G'} + |Q|) = \lambda\, \sigma_G$. Hence, since 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. □

Procedure 1 NC2REDUCTION (G)

$G' \leftarrow G$, i.e., $(V', E') \leftarrow (V, E)$;
$Q \leftarrow \emptyset$;
while (`true`)
  compute from G' the bipartite graph H and sets Y and Z;
  if ($Y \neq \emptyset$)
    $G' \leftarrow (V' - Y, E'[V' - Y])$; $Q \leftarrow Q \cup Y$;
  else if ($Z \neq V'$)
    $G' \leftarrow (Z, E'[Z])$;
  else
    return G' and Q;
  endif;
endwhile.

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 $G = (V, E)$, 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 $\hat{m} = |E|$. The rows will be indexed by the elements in $V \cup E$ and the columns will be indexed by the elements in $V \cup \{s\}$, where $s \notin V$ is a special symbol. The entries are defined by the following equations.

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

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

Proof: For every $v \in V$, denote by h_v the haplotype which is everywhere 1 except in column v, where $h_v[v] = 0$. Let X be a node cover of G. Consider the family $\mathcal{H}_X$ which contains as haplotypes the $\hat{n}$ genotypes (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 M. □

Lemma 8. *Let $\mathcal{H}$ be an optimal haplotype family explaining M and such that the number of 0's in haplotypes in $\mathcal{H}$ is minimum 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 which is everywhere 1 except in column v, where $h_v[v] = 0$. Let $\mathcal{H}$ be an optimal haplotype family explaining M. Then $\mathcal{H}$ contains as haplotypes all the $\hat{n}$ genotypes (rows) of M indexed by elements in V, since these do not contain any "2" symbol. At this point we can observe that any single genotype in M is either indexed by V (and hence is already explained), or is indexed by E (and hence can be explained by introducing a single haplotype of the form h_v, where $v \in E$. Therefore, the lemma follows once we have proven the following claim.

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

Proof: Clearly, h contains at most three 0's, otherwise h would not be good for any genotype in M. Actually, h contains at most two 0's, since otherwise h would be good for at most one single genotype in M indexed by E (we can clearly assume that G is simple). The same argument actually shows that if h has precisely two 0's, than one 0 is in the column indexed by s, and assume that the other 0 is in the column indexed by node v. But then all genotypes in M indexed by E and compatible with h would all correspond to edges incident at v and would hence be all explained by h_v plus the genotypes indexed by V which are anyhow in $\mathcal{H}$. Hence, $\mathcal{H} \setminus \{h\} \cup \{h_v\}$ would also explain 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 combine only with haplotypes h' which have precisely three 0's, two of which in correspondence of 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 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, since otherwise this haplotype would combine only with haplotypes h' which have precisely two 0's in correspondence of 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 $\mathcal{H}$ if we also replace each such h' in $\mathcal{H}$ with a haplotype h_v, with v 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), would follow an approximation algorithm with arbitrarily good performance guarantee for NC2, a contradiction.

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

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

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

where the third inequality follows from the fact that $\hat{n}/2 \leq \mathrm{Opt}$.

□

4 Integer Linear Programming formulations

In this section, we describe two alternative Integer Linear Programming (ILP) formulations for the problem (PPH). These formulations can be used to derive exact algorithms, by means, e.g., of some popular ILP solvers, such as CPLEX. The first formulation has an exponential number of variables and constraints, and is described also in Gusfield [9], where it was shown that it can be used for the practical solution of instances of (PPH). We review this formulation in Section 4.1, and we will use it in Section 5 to obtain an approximation algorithm for (PPH).

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

4.1 An exponential-size ILP formulation

Denote by $\hat{\mathcal{H}}_{\mathcal{G}}$ the set of those haplotypes which are compatible with some genotype in $\mathcal{G}$. We associate a $0-1$ variable x_h to every $h \in \hat{\mathcal{H}}_{\mathcal{G}}$. The intended meaning of $x_h = 1$ is that h is taken into the solution, whereas if $x_h = 0$ then h is not taken into the solution. Fix any total order on $\hat{\mathcal{H}}_{\mathcal{G}}$ and denote by $\mathcal{P}$ the set of those pairs (h_1, h_2) with $h_1, h_2 \in \hat{\mathcal{H}}_{\mathcal{G}}$, $h_1 < h_2$. 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 $\mathcal{P}_g := \{(h_1, h_2) \in \mathcal{P} \,|h_1 \oplus h_2 = g\}$.

The following is then a valid ILP formulation of the HAPLOTYPING problem:

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

subject to

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

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

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

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

Because of constraints (2), each genotype is explained by at least a pair of haplotypes. Constraints (3) and (4) insure that, unless both haplotypes h_1 and h_2 are taken in the solution, we cannot explain a genotype g with the pair (h_1, h_2).

4.2 A polynomial-size ILP formulation

As mentioned in Section 2.1, the input of the problem can be thought of as a matrix M, with entries in $\{0, 1, 2\}$, with m rows (the genotypes), while the output is a binary matrix M' with $2m$ rows (the haplotypes), and with the minimum number of distinct rows. For each row g of M (genotype) denote with 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) based on choosing, for each ambiguous position i of a genotype g, who between $g'[i]$ and $g''[i]$ is set to 0 and who is set to 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 $\mathrm{amb}(g) = \mathrm{amb}(g') = \mathrm{amb}(g'') := \{i \mid g[i] = 2\}$, the set of ambiguous positions of g. For each ambiguous $g \in \mathcal{G}$ and position $i \in \mathrm{amb}(g)$ we then define two 0-1 variables, i.e.

$$x^i_{g'}, x^i_{g''}.$$

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

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

Now we need variables and constraints to count how many distinct rows there are in M'. For simplicity, we first describe the model with more variables than really needed, and then discuss how to reduce their number.

Let $\mathcal{P}'$ be the set of pairs (a, b) such that $a, b \in \hat{\mathcal{G}}$ and $a < b$ For each two rows $(a, b) \in \mathcal{P}'$ 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

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

In order 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 $P(u, v) = u\,v$. This relation between $P(u, v)$, u, and v, can be obtained by forcing the following set, which we call $C(u, v)$, of linear constraints:

$$[C(u,v):] \qquad P(u,v) \leq u; \qquad P(u,v) \leq v; \qquad P(u,v) \geq u + v - 1. \tag{8}$$

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

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

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

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

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

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

These constraints are explained as follows. When i is not an ambiguous position for neither a nor b, y^i_{ab} is in fact a constant, as in constraints (9) and (10). If i is ambiguous for only one of a and b, e.g., for a, then 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 force the intended meaning for y_{ab} and z_r. These are:

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

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

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

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 other row identical to it which follows 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

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

with variables

$$x^i_a, y^j_{ab}, y_{ab}, z_a \in \{0, 1\} \qquad \forall (a, b) \in \mathcal{P}',\ i \in \text{amb}(a),\ j \in \{1, \ldots, n\}. \tag{18}$$

It is easy to see that there is a polynomial number of variables. In particular, there are $O(mn)$ variables x^i_a and $P(x^i_a, x^i_b)$, $O(m^2 n)$ variables y^i_{ab}, $O(m^2)$ variables y_{ab}, and $O(m)$ variables z_a. As for the constraints, there are $O(mn)$ constraints (6) and (17), $O(m^2 n)$ constraints between (9)-(13), $O(m^2 n)$ constraints (14), $O(m^2)$ constraints (15), and $O(m)$ constraints (16).

Overall, the model has $O(m^2 n)$ variables and $O(m^2 n)$ constraints. It is easy to reduce the number of both, by just observing that the decisions variables should be defined 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 non-ambiguous 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 could possibly be made identical in the solution. Then, we only need to define variables y_{ab} and y^i_{ab} for $(a,b) \in \mathcal{R}$ and i 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^i_{ab} (since, 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^i_a. Simple computations show that we obtain a model with a total of $O(Km)$ variables and constraints. Note that, on average, R can be much smaller than m^2 and K can be a small fraction of mn (at most 1/2. How small K is 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, on average, at that SNP).

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 Section 4.1.

Assume that every genotype $g \in \mathcal{G}$ contains at most k symbols "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^*) denote the optimal LP solution. Assume now to perform the following two actions, in their sequential order, on the solution (x^*, y^*):

1. scale up 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, since for any genotype g it is $|\mathcal{P}_g| \leq 2^{k-1}$, then for at least one pair $(h_1, h_2) \in \mathcal{P}_g$ it must be $y^*_{(h_1,h_2)} \geq 1/2^{k-1}$. 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 $x^*_{h_1} \geq 2^{k-1}$ and $x^*_{h_2} \geq 2^{k-1}$. But then, $\bar{x}^*_{h_1} = 1$ and $\bar{x}^*_{h_2} = 1$, so that constraints (3)-(4) are satisfied by $(\bar{x}, \bar{y})$.

Since for each h, $\bar{x}_h \leq 2^{k-1} x^*_h$, the value of the objective function scales up 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 $2^{k-1}\, z_{\mathrm{LP}} \leq 2^{k-1}\, \mathrm{OPT}$, i.e., a 2^{k-1}-approximation.

6 A randomized approximation algorithm

In this section we propose a randomized approximation algorithm for the problem (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 Section 5. On the other hand, there are reasons why the approach of this section can be preferred over the one of Section 5 in some cases. In particular, the approach described here is very simple to implement and does not require a third-party Linear Programming solver (independence from an LP-solver is usually desirable for Computational Biology software tools, so that people can freely download them and use them on their systems).

We employ the mathematical symbol log for base 2 logarithms and reserve the mathematical symbol ln to logarithms on the natural base e. We remind 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, which returns almost surely (i.e. with probability at least $\frac{m-1}{m}$) a feasible solution $\mathcal{H}_{feas}$ with $|\mathcal{H}_{feas}| \leq 2^{k+1}\lfloor \log m \rfloor (1 + \lceil \ln m \rceil)\,\mathrm{OPT}$. 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 $\mathcal{G}$ must be explained by haplotypes that are also used to explain other genotypes in $\mathcal{G}$. But 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 can then remove all these genotypes from $\mathcal{G}$. The mainstream bulk of this idea is developed and explained in full detail in Subsection 6.1. But there is a subtle obstacle which seems to prevent us from putting this simple idea in action: the suitable number of genotypes we better pick at random is a function of m, which we know, but also of OPT, which we do not know! To overcome this difficulty, we will first obtain a weaker result in which we assume the existence of an oracle function $\textsc{oracle}\mathrm{OPT}$ such that, when a call to $\textsc{oracle}\mathrm{OPT}(\mathcal{G})$ is invoked, the value of OPT for the genotype family $\mathcal{G}$ is returned. So, in order to prove Theorem 10, the following partial result will be first derived in Subsection 6.1.

Theorem 11 (Las Vegas with oracle). *Assume given an oracle $\textsc{oracle}\mathrm{OPT}(\mathcal{G})$ which, given an instance $\mathcal{G}$ of (PPH), returns the value $\mathrm{OPT} = \mathrm{OPT}(\mathcal{G})$ specifying the minimum number of haplotypes needed to explain $\mathcal{G}$. Then, there is a randomized algorithm which returns a feasible solution $\mathcal{H}_{feas}$ with $|\mathcal{H}_{feas}| \leq 2^{k+1}\lfloor \log m \rfloor (1 + \lceil \ln m \rceil)\,\mathrm{OPT}$.*

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.

Since the average number of genotypes which are compatible with the same haplotype of the solution depends both on m and on OPT, in the proof of Theorem 11 it is assumed that the value OPT is known. To obtain a true approximation algorithm as announced in Theorem 10, we must remove the dependence of the result from the knowledge of the actual value of OPT. How this is done is explained in Subsection 6.2, where the full Monte Carlo algorithm of Theorem 10 is finally 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 $\mathcal{G} = \emptyset$ then return $\emptyset$;
2. OPT $\leftarrow$ ORACLEOPT($\mathcal{G}$);
3. repeat
4.   TEMP $\leftarrow \emptyset$;
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 $\frac{1}{2}$ of the genotypes in $\mathcal{G}$;
10. $\mathcal{G}' \leftarrow \{g \in \mathcal{G} \ : g$ is not explained by TEMP$\}$;
11. return TEMP $\cup$ HAPLOTYPE($\mathcal{G}'$).

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

Lemma 12. *When Algorithm 2 terminates, then it returns a haplotype family $\mathcal{H}_{feas}$ which explains $\mathcal{G}$. Moreover, $|\mathcal{H}_{feas}| \leq 2^{k+1} \lfloor \log m \rfloor \,(1 + \lceil \ln m \rceil)\,$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, $|\mathcal{G}'| \leq \frac{1}{2}|\mathcal{G}| < |\mathcal{G}|$ by the test at step 9. Therefore, we can assume by induction that HAPLOTYPE($\mathcal{G}'$) returns a set of haplotypes which explains $\mathcal{G}'$. Since TEMP explains all genotypes in $\mathcal{G} \setminus \mathcal{G}'$ by the way

$\mathcal{G}'$ is defined at step 10, then HAPLOTYPE($\mathcal{G}$) returns a haplotype family which explains all genotypes in $\mathcal{G}$. This happens at step 11, where TEMP $\cup$ HAPLOTYPE($\mathcal{G}'$) is returned. To prove the approximation guarantee, we have to bound the size of TEMP $\cup$ HAPLOTYPE($\mathcal{G}'$). By induction, the size of HAPLOTYPE($\mathcal{G}'$) is bounded by $2^{k+1}\lfloor \log \lfloor \frac{m}{2} \rfloor \rfloor (1 + \lceil \ln m \rceil)\,\mathrm{OPT}$, since $|\mathcal{G}'| \leq \frac{1}{2}|\mathcal{G}| = \frac{m}{2}$ and since any haplotype family which 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. Since TEMP is each time reset to an empty set at step 4, and since in the iterations 5–8 we choose at most $2\,(1 + \lceil \ln m \rceil)\,\mathrm{OPT}$ different genotypes from $\mathcal{G}$, then the number of haplotypes in TEMP never exceeds $2^k\, 2\,(1 + \lceil \ln m \rceil)\,\mathrm{OPT} = 2^{k+1}\,(1 + \lceil \ln m \rceil)\,\mathrm{OPT}$. Therefore,

$$
\begin{aligned}
|\mathcal{H}_{feas}| &= |\textsc{Temp} \cup \textsc{Haplotype}(\mathcal{G}')| \leq |\textsc{Temp}| + |\textsc{Haplotype}(\mathcal{G}')| \\
&\leq 2^{k+1}\,(1 + \lceil \ln m \rceil)\,\mathrm{OPT} + 2^{k+1}\lfloor \log \lfloor \frac{m}{2} \rfloor \rfloor\,(1 + \lceil \ln m \rceil)\,\mathrm{OPT} \\
&\leq 2^{k+1}\lfloor 1 + \log \lfloor \frac{m}{2} \rfloor \rfloor\,(1 + \lceil \ln m \rceil)\,\mathrm{OPT} = 2^{k+1}\lfloor \log m \rfloor\,(1 + \lceil \ln m \rceil)\,\mathrm{OPT}.
\end{aligned}
$$

□

We are now left with the analysis of the running time. This will take the remainder of this subsection and will motivate the design of the Monte Carlo algorithm introduced in Subsection 6.2.

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

To better understand the following analysis, it is useful to think of a graph on node set $\hat{\mathcal{H}}_{\mathcal{G}}$ 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 $h \in \hat{\mathcal{H}}_{\mathcal{G}}$. We define a node in $\mathcal{H}_{opt}$ *to be good* when the fraction of genotypes in $\mathcal{G}$ compatible with h is at least $\frac{1}{2\mathrm{OPT}}$.

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

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

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

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

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

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

□

For a second motivation, consider the following observation.

Observation 15. *When h is good, then the fraction of those genotypes in $\mathcal{G}$ which are compatible with h is at least $\frac{1}{2\text{OPT}}$.*

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

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

□

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

Lemma 16. *Each time we get to step 9 we have that $P[\mathcal{H}_{opt}^{good} \subseteq \text{TEMP}] \geq \frac{e-2}{e}$.*

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 we have that $P[h \notin \text{TEMP}] \leq \frac{1}{em}$. From this it will easily follow that the probability that any good haplotype in $\mathcal{H}_{opt}$ is not in TEMP is at most $\frac{2}{e}$.

Since h is good, and since the random choices are independent, then

$$\begin{aligned} P[h \notin \text{TEMP}] &\leq \textstyle\prod_{i=1}^{2(1+\lceil \ln m \rceil)\text{OPT}} \left(1 - \frac{1}{2\text{OPT}}\right) = \left(1 - \frac{1}{2\text{OPT}}\right)^{2(1+\lceil \ln m \rceil)\text{OPT}} = \\ &= \left(\left(1 - \frac{1}{2\text{OPT}}\right)^{2\text{OPT}}\right)^{1+\lceil \ln m \rceil} \leq \left(\frac{1}{e}\right)^{1+\lceil \ln m \rceil} \leq \frac{1}{em} \end{aligned}$$

To bound the probability that any good haplotype does not make it into TEMP, we employ the most 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 are the dependencies between these events.

Hence, where $\mathcal{H}_{opt}^{good}$ is the set of good haplotypes in $\mathcal{H}_{opt}$, then

$$\begin{aligned} P[\exists h \in \mathcal{H}_{opt}^{good} \setminus \text{TEMP}] &\leq \textstyle\sum_{h \in \mathcal{H}_{opt}^{good}} P[h \notin \text{TEMP}] \leq \\ &\leq \textstyle\sum_{h \in \mathcal{H}_{opt}^{good}} \frac{1}{em} = \frac{|\mathcal{H}_{opt}^{good}|}{em} \leq \frac{|\mathcal{H}_{opt}|}{em} = \frac{OPT}{em} \leq \frac{2}{e}. \end{aligned}$$

Where 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 to pass the test and exit the repeat-loop is at least $\frac{e-2}{e}$.*

Proof: By Lemma 16, each time the test at step 9 is faced we have that $P[\mathcal{H}_{opt}^{good} \subseteq \textsc{Temp}] \geq \frac{e-2}{e}$. By Lemma 14, this means that the probability that the test is positive is at least $\frac{e-2}{e}$. □

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 to prove this formally, since these arguments will not enter in 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 simply to perform a binary search on OPT at each recursion of Algorithm 2. To do so, we need to introduce a procedure which tests a candidate value for OPT. We call this procedure TryOPTvalue. Procedure TryOPTvalue receives as input a genotype family $\mathcal{G}$, an hypothesis value OPT′ for OPT, and a parameter numAttempts. If procedure TryOPTvalue succeeds in finding a haplotype family which explains at least half of $\mathcal{G}$ under the hypothesis value OPT′ for OPT and 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)

---

for $t := 1$ to numAttempts do
    Temp $\leftarrow \emptyset$;
    for $2\,(1 + \lceil \ln m \rceil)\,\mathrm{OPT}'$ times do
        choose uniformly at random a $g \in \mathcal{G}$;
        create, and put in Temp, all haplotypes compatible with g;
    endfor;
    if Temp explains at least $\frac{1}{2}$ of the genotypes in $\mathcal{G}$ then
        return Temp;
endfor;
return NO;

With respect to the above procedure, we can 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 $(\mathcal{G}, \mathrm{OPT}', e \ln m)$ *returns a NO answer is at most* $\frac{1}{m^2}$.

Proof: By Corollary 17, each time the test at step 7 is faced, the probability of success is at least $\frac{e-2}{e}$. Therefore, the probability that TRYOPTVALUE $(\mathcal{G}, \mathrm{OPT}', e \ln m)$ returns a NO answer is bounded by

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

□

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. 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 which are compatible with a given genotype g requires $O(2^k n) = O(n)$ time. □

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

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

---

if $\mathcal{G} = \emptyset$ then return $\emptyset$;
$\mathrm{OPT}_1 \leftarrow \sqrt{m}$; $\mathrm{OPT}_2 \leftarrow 2m$; $\mathcal{H}_{temp} \leftarrow$ DOUBLE$(\mathcal{G})$;
while $\mathrm{OPT}_2 > \mathrm{OPT}_1 + 1$ do
    $\mathrm{OPT}' \leftarrow \lfloor \frac{\mathrm{OPT}_1 + \mathrm{OPT}_2}{2} \rfloor$;
    TEMP $\leftarrow$ TRYOPTVALUE $(\mathcal{G}, \mathrm{OPT}', e \ln m)$;
    if TEMP is a NO answer then $\mathrm{OPT}_1 \leftarrow \mathrm{OPT}'$;
    else set $\mathrm{OPT}_2 \leftarrow \mathrm{OPT}'$ and $\mathcal{H}_{temp} \leftarrow$ TEMP;
endwhile;
$\mathcal{G}' \leftarrow \{g \in \mathcal{G} : g$ is not explained by $\mathcal{H}_{temp}\}$;
return $\mathcal{H}_{temp} \cup$ MCHAPLOTYPE$(\mathcal{G}')$.

Remark 20. *Algorithm 4 is an almost linear time algorithm. More precisely, where $m = |\mathcal{G}|$ and n is the number of SNPs, then 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 $O(m \log^3 m(n + \log m))$ since TRYOPTVALUE $(\mathcal{G}, \mathrm{OPT}', e \ln m)$ is called $O(\log m)$ times. This dominates the time for the whole execution since m halves at each recursion. □

The next lemma closes our analysis of Algorithm 4.

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

Proof: It should be clear that MCHAPLOTYPE($\mathcal{G}$) always explains $\mathcal{G}$. Moreover, we know that MCHAPLOTYPE($\mathcal{G}$)$| \leq 2^{k+1}\lfloor \log m \rfloor (1 + \lceil \ln m \rceil)\,\mathrm{OPT}$ holds if during the whole execution of the algorithm the following situation has never occurred.

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

Note that, during the execution of MCHAPLOTYPE($\mathcal{G}$), 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 $\frac{1}{m^2}$. Therefore,

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

□

7 Conclusions

In this paper we have taken a comprehensive approach to the study of the Pure Parsimony Haplotyping problem. Our results have concerned the three main aspects of the problem, i.e. (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 polynomially solvable then?

(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 have given 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 which 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, what seems to be the most interesting question about approximation algorithms is: can we design a constant δ-approximation algorithm for the (PPH) problem?

References

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

[2] V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph. Haplotyping as perfect phylogeny: a direct approach. *Journal of Computational Biology*, 10(3-4):323–340, 2003.

[3] P. Berman and M. Karpinski. On some tighter inapproximability results. ECCC report no. 29, University of Trier, 1998.

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

[5] A. Clark. Inference of haplotypes from PCR–amplified samples of diploid populations. *Molecular Biology Evolution*, 7:111–122, 1990.

[6] E. Eskin, E. Halperin, and R. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. *Journal of Bioinformatics and Computational Biology*, 1(1):1–20, 2003.

[7] D. Gusfield. A practical algorithm for optimal inference of haplotypes from diploid populations. In R. Altman, T.L. Bailey, P. Bourne, M. Gribskov, T. Lengauer, I.N. Shindyalov, L.F. Ten Eyck, and H. Weissig, editors, *Proceedings of the Annual International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 183–189, Menlo Park, CA, 2000. AAAI Press.

[8] D. Gusfield. Inference of haplotypes from samples of diploid populations: Complexity and algorithms. *Journal of Computational Biology*, 8(3):305–324, 2001.

[9] D. Gusfield. Haplotype inference by pure parsimony. In *Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 2676 of *Lecture Notes in Computer Science*, pages 144–155. Springer, 2003.

[10] L. Helmuth. Genome research: Map of the human genome 3.0. *Science*, 293(5530):583–585, 2001.

[11] E. Hubbel. Personal communication. 2002.

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

[13] G. Lancia, V. Bafna, S. Istrail, R. Lippert, and R. Schwartz. SNPs problems, complexity and algorithms. In *Proceedings of the Annual European Symposium on Algorithms (ESA)*, volume 2161 of *Lecture Notes in Computer Science*, pages 182–193. Springer, 2001.

[14] L. Li, J. H. Kim, and M. Waterman. Haplotype reconstruction from SNP alignment. In *Proceedings of the Annual International Conference on Computational Molecular Biology (RECOMB)*, New York, NY, 2003. ACM Press.

[15] R. Lippert, R. Schwartz, G. Lancia, and S. Istrail. Algorithmic strategies for the SNPs haplotype assembly problem. *Briefings in Bioinformatics*, 3(1):23–31, 2002.

[16] E. Marshall. Drug firms to create public database of genetic mutations. *Science Magazine*, 284(5413):406–407, 1999.

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

[18] V. Kann P. Alimonti. Hardness of approximating problems on cubic graphs. In *Proceedings of 3rd Italian Conference on Algorithms and Complexity (ICALP)*, volume 1203 of *Lecture Notes in Computer Science*, pages 288–298. Springer, 1997.

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

[20] R. Rizzi, V. Bafna, S. Istrail, and G. Lancia. Practical algorithms and fixed-parameter tractability for the single individual SNP haplotyping problem. In R. Guigo and D. Gusfield, editors, *Proceedings of Annual Workshop on Algorithms in Bioinformatics (WABI)*, volume 2452 of *Lecture Notes in Computer Science*, pages 29–43. Springer, 2002.

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