The phasing of heterozygous traits: Algorithms and complexity

Giuseppe Lancia

Dipartimento di Matematica e Informatica, Università di Udine, Via delle Scienze 206, 35100 Udine, Italy

Abstract

Combinatorial haplotyping problems have received great attention in the past few years. We review their definitions and the main results that were obtained for their solution. Haplotyping problems require one to determine a set $H$ of binary vectors (called haplotypes) that explain a set of $G$ of ternary vectors (called genotypes). The number $\chi(G)$ of haplotypes to choose from can be exponential with respect to the number of genotypes. We give an exact formula, based on the inclusion–exclusion principle, for determining $\chi(G)$.

© 2007 Elsevier Ltd. All rights reserved.

Keywords: Computational biology; Pure parsimony haplotyping; Perfect phylogeny haplotyping; Clark’s rule

1. Introduction

A polymorphism is a common trait that shows a statistically significant variability within a population. Some polymorphisms can be readily observed at the phenotype level (e.g., the eye color) while others are more “hidden” (e.g., the blood type), but, ultimately, all polymorphisms find an explanation at genomic level: they are variations within the DNA sequence of a species. There can be polymorphic genomic regions of arbitrary length. The smallest possible length is just one nucleotide. Such polymorphic nucleotide positions are called Single Nucleotide Polymorphisms (or, shortly, SNPs, pronounced “snips”). By far, most human polymorphisms are in fact SNPs, occurring, on average, one in 500 bases [1].

The values that can be observed at a given SNP in a population are called alleles. Almost all SNPs are bi-allelic, i.e., out of the four nucleotides A, C, T, G, only two are observed at any SNP (not necessarily the same two for different SNPs). To be a SNP, a site must show a statistically significant variability, with the least frequent allele present in at least 5% of the population.

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. As a consequence, alleles are inherited in sets, the set of paternal alleles contained on the paternal chromosome copy and the set of maternal alleles, on the maternal chromosome copy. The sequence of alleles on a chromosome copy is called a haplotype. For a given SNP, an individual can be either homozygous, if both parents contributed the same allele, or heterozygous, if the paternal and maternal alleles are different.

E-mail address: lancia@dimi.uniud.it.

0898-1221/$ - see front matter © 2007 Elsevier Ltd. All rights reserved. doi:10.1016/j.camwa.2006.12.089
In Fig. 1, we illustrate a simplistic example of three individuals and four SNPs. The alleles for SNP 1, in this example, are C and G. Individual 1, in this example, is heterozygous for SNPs 1, 2 and 3, and homozygous for SNP 4. Individual 1’s haplotypes are CCCT and GAGT.

Haplotyping an individual consists in determining that individual’s two haplotypes, for a given chromosome. Haplotyping a population consists in haplotyping each individual of the population. With the larger availability in SNP data, recent years have seen the birth of a set of new computational problems related to haplotyping. These problems are motivated by the fact that it is today economically infeasible to determine the haplotypes by a “wet-lab” experiment. On the other hand, there is a cheap experiment which can determine the (less informative and often ambiguous) genotypes, from which the haplotypes must then be retrieved \textit{in silico}, i.e., computationally.

A genotype of an individual contains the information about the two – possibly identical – alleles at each SNP, but without specifying their origin (paternal or maternal). Given a genotype, there may be many possible pairs of haplotypes that justify that genotype. For example, assume we only know that individual 1 in Fig. 1 is heterozygous for the alleles \{C, G\} at SNP 1 and for the alleles \{A, C\} at SNP 2. Then, either one of these alternatives may be true:

(i) One parent has contributed the alleles C and A and the other the alleles G and C
(ii) One parent has contributed the alleles C and C and the other the alleles G and A.

Both possibilities are plausible. Associating the alleles to the parents is called phasing the alleles. For k heterozygous SNPs there are \(2^k\) possible phasings, which makes choosing the correct one a difficult problem. Once the alleles have been phased the two haplotypes are inferred, so that phasing and haplotyping are in fact the same problem. The two haplotypes that are obtained by phasing the alleles are said to resolve, or to explain the genotype.

In a general way, a population haplotyping problem can be stated as follows:

Given a set \(G\) of genotypes, corresponding to an existing, unknown, set \(H\) of haplotypes, retrieve \(H\).

Another way to formulate the problem is:

Given a set \(G\) of genotypes, compute the set \(H\) of haplotypes which contains, for each genotype \(g\), the two haplotypes \(h_1, h_2\) obtained by the correct phasing of \(g\).

It is not easy to describe constraints, based only on the knowledge of \(G\), that define precisely which of the exponentially many phasings of a genotype is the correct one. Biologists have therefore described several sensible criteria for “good” phasings. For instance, under a commonly accepted parsimony principle (supported by the fact that haplotypes could be traced back in time to a small set of ancestors from which we all descend), a good solution may be one which minimizes the number of distinct haplotypes inferred.

Once it has been mathematically modeled, haplotyping gives rise to several nice and challenging combinatorial problems \([2, 3]\). These problems, have been extensively studied in the last few years. Some of them have been proven NP-hard and solved by (worst-case) exponential-time algorithms, while for others, polynomial-time algorithms have been found.

In this survey, we address some of the most interesting haplotyping models that have been proposed in the literature, such as \textit{Clark’s rule} \([4, 5]\), \textit{Pure parsimony} \([6–8]\), \textit{Perfect phylogeny} \([9, 10]\), and \textit{Disease association} \([11]\). Each model and objective function has specific biological motivations, which are discussed in the following sections.

Here, we focus on the combinatorial approach to haplotyping problems. It should be remarked that there is also a very important \textit{statistical} approach to haplotyping problems, which does not fall within the scope of this survey. The statistical approach has led to widely used software tools for haplotyping, such as the program PHASE \([12]\).
1.1. Notation and definitions

Given a set of $n$ SNPs, fix arbitrarily a binary encoding of the two alleles for each SNP (i.e., call one of the two alleles 0 and the other 1). Once the encoding has been fixed, each haplotype corresponds to (with a slight abuse of terminology, hereafter, we will say that each haplotype is) a binary vector of length $n$.

For a haplotype $h$, we denote by $h[i]$ the value of its $i$-th component, with $i = 1, \ldots, n$. Given two haplotypes $h'$ and $h''$, their sum is defined as a vector $g := h' \oplus h''$. The vector $g$ has length $n$, and its components can take only values in $\{0, 1, 2\}$, according to the following rule:

$$
g[i] := \begin{cases} 
0 & \text{if } h'[i] = h''[i] = 0 \\
1 & \text{if } h'[i] = h''[i] = 1 \\
2 & \text{if } h'[i] \neq h''[i].
\end{cases}
$$

We call a vector $g$ with entries in $\{0, 1, 2\}$ a genotype. Each position $i$ such that $g[i] = 2$ is called an ambiguous position (or ambiguous site). We denote by $A(g) \subseteq \{1, \ldots, n\}$ the set of ambiguous positions of $g$. Biologically, genotype entries of value 0 or 1 correspond to homozygous SNP sites, while entries of value 2 correspond to heterozygous sites. In Fig. 2 we illustrate a case of three individuals, showing their haplotypes and genotypes.

A resolution of a genotype $g$ is given by a pair of haplotypes $h'$ and $h''$ such that $g = h' \oplus h''$. Such haplotypes are said to resolve $g$. A genotype is ambiguous if it has more than one possible resolution, i.e., if it has at least two ambiguous positions. A haplotype $h$ is said to be compatible with a genotype $g$ if $h$ can be used in a resolution of $g$. It is immediate to see that $h$ is compatible with $g$ if and only if at each position where $g[i] \neq 2$ it is $g[i] = h[i]$. Two genotypes $g$ and $g'$ are compatible if there exists at least one haplotype compatible with both of them, otherwise, they are incompatible. It is immediate to see that $g$ and $g'$ are compatible if and only if at each position $i$ where they are both non-ambiguous, it is $g[i] = g'[i]$.

For the haplotyping problems described in this article, the input data consist in a set $G$ of $m$ genotypes $g_1, \ldots, g_m$, corresponding to $m$ individuals in a population. The output is set $H$ of haplotypes such that, for each $g \in G$, there is at least one pair of haplotypes $h', h'' \in H$ with $g = h' \oplus h''$. Such a set $H$ of haplotypes is said to explain $G$. In addition to explaining $G$, the set $H$ is also required to satisfy some particular constraints. These constraints are different for different specific types of haplotyping problems. For each problem described in this survey, the particular constraints are given in the corresponding section.

2. Counting the haplotypes

Let us consider the following problem: given a set of genotypes $G$, how many distinct haplotypes exist that are compatible with genotypes of $G$? Let us call this number $\chi(G)$. Assume $|G| = m$ and there are $n$ SNPs. Then, the

<table>
<thead>
<tr>
<th>Haplotype 1, paternal:</th>
<th>0 1 0 1</th>
<th>2 2 2 1</th>
<th>Genotype 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haplotype 1, maternal:</td>
<td>1 0 1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>Haplotype 2, paternal:</td>
<td>0 0 1 1</td>
<td>2 2 1 2</td>
<td>Genotype 2</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>Haplotype 2, maternal:</td>
<td>1 1 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>Haplotype 3, paternal:</td>
<td>0 0 1 1</td>
<td>2 0 2 2</td>
<td>Genotype 3</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>Haplotype 3, maternal:</td>
<td>1 0 0 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2. Haplotypes and corresponding genotypes.

The remainder of the paper is organized as follows. In Section 1.1 we introduce a mathematical notation for haplotyping problems. In Section 2 we describe a counting argument to determine the exact number of haplotypes that can be inferred from a given set of genotypes. In Section 3 we survey some combinatorial models for population haplotyping. In particular, in Section 3.1 we describe Clark’s rule; in Section 3.2 we discuss haplotyping by pure parsimony; in Section 3.3 we discuss haplotyping for perfect phylogeny; finally, in Section 3.4 we discuss haplotyping with respect to a genetic disease that affects some members of the population.
data can be also regarded as an \( m \times n \) matrix \( G = (g_1 \ldots g_m)^T \), where \( g_i[j] \) is the \( j \)-th entry in row \( i \).

There are two possible ways of computing \( \chi(G) \):

1. By exhaustively considering each \( g \in G \), generating all possible haplotypes compatible with it, and keeping the union of all haplotypes thus obtained.
2. By using the principle of inclusion–exclusion, as described below.

Which way is better depends on \( m, n \) and on the total number of 2s in \( G \). Let us consider the exhaustive approach first.

Let \( M_2 \) be the maximum number of 2s in any row of \( G \). Note that \( M_2 \) could be \( \Theta(n) \). Generating all haplotypes compatible with a row of \( G \) gives rise to \( O(2^{M_2}) \) haplotypes, of length \( n \) each. We need to store the haplotypes so that we know when one has already been generated. This can be achieved with data structures in which, for \( N \) elements, membership-test/insertion has cost \( O(\log N) \). Since for us \( N = O(2^n) \), this cost is \( O(n) \). We get a total cost of \( mn2^{M_2} \), i.e.,

\[
 mn2^n \tag{1}
\]

in the worst case. It is very important to realize that this method also requires memory in the order of \( O(2^n) \).

We now turn to the inclusion–exclusion approach. Given any subset \( X \subseteq G \), let us denote by \( a(X) \) the following quantity:

\[
a(X) := \begin{cases} -\infty & \text{if there exist incompatible } g, g' \in X \text{ or } X = \emptyset \\ |\bigcap_{g \in X} A(g)| & \text{otherwise.} \end{cases}
\]

Notice that \( a(X) \neq -\infty \) if and only if there is at least one haplotype compatible with each genotype of \( X \), and that, in this case, \( a(X) \) counts the number of positions in which all genotypes of \( X \) are ambiguous.

Let us denote by \( H_X \) the set of haplotypes that are compatible with all genotypes of \( X \) (i.e., \( h \in H_X \) if \( h \) is compatible with \( g \) for all \( g \) in \( X \)). The following lemma shows how to easily compute \( |H_X| \).

**Lemma 1.** Let \( X \subseteq G \). Then, where \( 2^{-\infty} \) is defined to be 0, it is \( |H_X| = 2^{a(X)} \).

**Proof.** If \( a(X) = -\infty \) then there are no haplotypes compatible with all \( X \), and \( |H_X| = 0 \). On the other hand, if \( a(X) \neq -\infty \), then at each position \( i \) which is non-ambiguous for at least a genotype \( g \in X \), a haplotype \( h \in H_X \) must have \( h[i] = g[i] \). The only free positions of \( h \) are the columns ambiguous for all genotypes, which can be fixed in \( 2^{a(X)} \) ways. \( \diamond \)

Now, we can give a formula for \( \chi(G) \):

**Theorem 1.**

\[
\chi(G) = \sum_{X \subseteq G} (-1)^{|X|-1}|H_X| = \sum_{X \subseteq G} (-1)^{|X|-1}2^{a(X)}.
\]

**Proof.** The formula follows from the notorious inclusion/exclusion principle [13]. First, all haplotypes generated by each single genotype are added up. But then, haplotypes generated by two genotypes were counted more than once, and so we subtract \( |H_X| \) for each \( X \) of size two. Again, a haplotype which can be generated by three genotypes was removed too many times, and so it is added up to the count again, and so on. \( \diamond \)

What is the cost of computing \( \chi(G) \) as described above? There are \( 2^m \) possible subsets \( X \). For a subset \( X \) of size \( k \), the cost of computing \( a(X) \) (from which \( |H_X| \) can be immediately derived) is \( O(kn) \). Since there are \( \binom{m}{k} \) subsets of size \( k \), the total cost is

\[
n \times \sum_{k=0}^{m} \binom{m}{k} k.
\]

We have

\[
\sum_{k=0}^{m} \binom{m}{k} k + \sum_{k=0}^{m} \binom{m}{k} (m - k) = m \sum_{k=0}^{m} \binom{m}{k} = m2^m
\]
and, since
\[ \sum_{k=0}^{m} \binom{m}{k} k = \sum_{k=0}^{m} \binom{m}{k} (m - k) \]
we have
\[ \sum_{k=0}^{m} \binom{m}{k} k = m \times 2^{m-1} \]
from which the total cost of computing the inclusion/exclusion count is
\[ mn2^{m-1}. \]

It is important to notice that the memory required to compute \( \chi(G) \) this way is linear in \( m \). In fact, it is enough to generate all subsets \( X \) incrementally, each time throwing away the previous one (this is akin to counting from 0 to \( 2^m - 1 \) in binary).

The basic conclusion is that, when \( m \) is smaller than \( n \), it is better to count by inclusion/exclusion. Other reasons for doing so even when \( m \) is bigger than \( n \) are the small memory required and the fact that the inclusion/exclusion count is easier to implement, not needing data structures to store the haplotypes.

In order to speed up the computation of \( \chi(G) \) by inclusion/exclusion, one can use the following divide-et-impera approach.

**Lemma 2.** Let \( j \) be a column of \( G \) where both 0s and 1s appear. Let \( Z, O \) and \( T \) be the subsets of rows (genotypes) that have a zero, one, or two, respectively, in column \( j \). Then
\[ \chi(G) = \chi(Z \cup T) + \chi(O \cup T) - \chi(T). \]

**Proof.** A haplotype generated from \( G \) can be generated from a genotype in \( Z \), \( O \) or \( T \). All genotypes generated from \( Z \) are counted by the above formula (first term), as well as those generated from \( O \) (second term) and there is no double-counting since genotypes in \( Z \) are not compatible with genotypes in \( O \). As for genotypes generated from \( T \), they are double-counted by the first two terms, so we subtract \( \chi(T) \) in the end. \( \diamond \)

The above approach has the potential to dramatically decrease the time needed to compute \( \chi(G) \). For instance, assume \( m = 30 \) and \( |Z| = |O| = |T| = 10 \). Then, loosely speaking, without the above decomposition the total time to compute \( \chi(G) \) is proportional to \( 2^{30} \), while with the decomposition, it becomes proportional to \( 2^{20} \). If \( |Z| = |O| = 15 \), the time is proportional to \( 2^{15} \). It is very important to note that the approach can be iterated recursively: i.e., each of the matrices \( O \cup T \), \( Z \cup T \) and \( T \) goes through the same process, by identifying in it a column showing both 0s and 1s, and decomposing it accordingly. The only time when the inclusion/exclusion principle should be applied without decomposition is when each column shows only, besides 2s, either 0s or 1s.

### 3. Combinatorial approaches to phasing

#### 3.1. Clark’s rule

In this section we review the problem of determining a set of haplotypes for \( G \) via the optimal application of an inference rule known as Clark’s rule. The rule, proposed by the geneticist Clark [14], can be used to resolve a genotype \( g \) whenever a haplotype \( h \) compatible with \( g \) is known. The rule derives new haplotypes by inference from known ones as follows:

**Clark’s inference rule:** Given a genotype \( g \) and a compatible haplotype \( h \), obtain a new haplotype \( q \) by setting
\[ q[j] := 1 - h[j] \] at all positions \( j \in A(g) \) and \( q[j] := h[j] \) at the remaining positions.

Notice that \( q \) and \( h \) are a resolution of \( g \). In order to resolve all genotypes of \( G \), Clark suggested the use of successive applications of the inference rule. Since the rule requires a “bootstrap” set of haplotypes used to derive new
haplotypes, the starting haplotypes are obtained by resolving, in the unique possible way, the unambiguous genotypes in $G$ (of which it is assumed there is always at least one).

In essence, Clark proposed the following algorithm, whose validity was supported by arguments from theoretical population genetics:

Clark’s algorithm: Let $G'$ be the set of non-ambiguous genotypes, and let $H$ be the set of haplotypes obtained from $G'$. Reset $G := G - G'$. Then, repeat the following. If they exist, take a $g \in G$ and a compatible $h \in H$ and apply the inference rule, obtaining $q$. Set $G := G - \{g\}$, $H := H \cup \{q\}$, and iterate. When no such $g$ and $h$ exist, the algorithm has succeeded if $G = \emptyset$ and has failed otherwise.

Notice that the “algorithm” is non-deterministic since it does not specify how to choose the pair $(g, h)$ whenever there are more candidates to the application of the rule. For example, suppose $G = \{2000, 2200, 1122\}$. The algorithm starts by setting $H = \{0000, 1000\}$ and $G = \{2200, 1122\}$. The inference rule can be used to resolve 2200 from 0000, obtaining 1100, which can, in turn, be used to resolve 1122, obtaining 1111. However, one could have started by using 1000 to resolve 2200 obtaining 0100. At that point, there would be no way to resolve 1122. The non-determinism in the choice of the pair $g, h$ to which the inference rule is applied can be settled by fixing a deterministic rule based on the initial sorting of the data. In [14], a large number of random sortings is used to run the algorithm, and the best solution overall is reported. Tests on real and simulated data sets showed that, although most times the algorithm could resolve all genotypes, many times the algorithm failed.

In order to apply Clark’s rule to its best, one should solve the following haplotyping problem:

[HAPLOTYPING FOR CLARK’S RULE]

INSTANCE: A set $G$ of genotypes.

PROBLEM: Find a set $H$ of haplotypes such that $H$ is obtained by successive applications of Clark’s rule and the number of genotypes in $G$ that have no resolution in $H$ is minimum.

The problem calls for the ordering of application of the inference rule that leaves the fewest number of unresolved genotypes in the end. This problem was first defined and studied by Gusfield [15], who proved it is NP-hard and APX-hard. (A problem is APX-hard if there is a constant $\alpha > 1$ such that the existence of an $\alpha$-approximation algorithm would imply $P = NP$. See [16] for a full description of the class APX.)

As for practical algorithms, Gusfield [4,5] proposed an integer programming approach for a graph-theoretic reformulation of the problem. The problem is first transformed (by an exponential-time reduction) into a problem on a digraph $(N, E)$, defined as follows. Let $N = \bigcup_{g \in G} N(g)$, where $N(g) := \{(h, g) : h \text{ is compatible with } g\}$. Let $N' = \bigcup_{g \in G'} N(g)$ be (isomorphic to) the subset of haplotypes determined from the set $G'$ of unambiguous genotypes. For each pair $v = (h, g')$, $w = (q, g)$ in $N$, there is an arc $(v, w) \in E$ if $g$ is ambiguous, $g' \neq g$, and $g = h \oplus q$ (i.e., $q$ can be inferred from $g$ via $h$). Then, a directed tree rooted at a node $v \in N'$ specifies a feasible history of successive applications of the inference rule starting at node $v \in N'$. The problem can then be stated as: find the largest number of nodes that can be reached by a set of node-disjoint directed trees, where each tree is rooted at a node in $N'$ and where for every ambiguous genotype $g$, at most one node in $N(g)$ is reached.

The above graph problem was shown to be NP-hard [15] (note that the reduction of the haplotyping problem to this one is exponential-time, and hence it does not imply NP-hardness trivially). For its solution, Gusfield proposed an integer programming formulation, and noticed that the solution of the linear programming relaxation was very often integer for the real-life instances tested. The model was applied to real data as well as random instances, with up to 80 genotypes, of which 60 ambiguous, over 15 SNPs.

3.2. Parsimony

Since Clark’s algorithm tries to re-use existing haplotypes and to introduce new haplotypes only when strictly needed, the solution it tends to produce is usually of small size. Notice that the maximum size for a set $H$ resolving $G$ is $2|G|$, while the smallest possible is $\Omega(\sqrt{|G|})$.

The explicit objective of minimizing the size of $H$ has several biological motivations. For one, the number of distinct haplotypes observed in nature is vastly smaller than the number of possible haplotypes. As we all descend from a small number of ancestors, their haplotypes, modulo some recombination events and mutations, must be the
same we possess today. Finally, under a general parsimony principle (also known as Occam’s razor), of many possible explanations of an observed situation, one should favor the simplest.

Therefore, the following pure parsimony haplotyping problem has been defined and studied:

**[HAPLOTYPING FOR PURE PARSIMONY]**

**INSTANCE:** A set \( G \) of genotypes.

**PROBLEM:** Find a set \( H \) of haplotypes such that \( H \) resolves \( G \) and the cardinality of \( H \) is minimum.

This problem has been introduced by Gusfield [6], who adopted an integer programming formulation for its practical solution. The problem is NP-hard, as first shown by Hubbel [17]. Lancia et al. [7] show that, in fact, the problem is APX-hard, even when each genotype is restricted to possess at most three ambiguous sites. In the case where each genotype has at most two ambiguous sites, the problem is polynomial (Lancia and Rizzi [18]). Although the problem is APX-hard, there exist constant-ratio approximation algorithms under the restriction that each genotype has at most \( k \) ambiguous sites [7].

Pure parsimony haplotyping has been attacked by means of several mathematical programming approaches. In particular, there have been

- **Integer programming formulations of exponential size.** The first such formulation was given by Gusfield [6], and has \( \chi(G) \) variables and \( O(m2^n) \) constraints. In [6] the experimental results show that this model can be used to tackle problems with up to 50 genotypes, over 30 SNPs, with relatively small levels of heterozygosity.

- **Integer programming formulations of polynomial size and hybrid formulations.** Many authors (most prominently, Brown and Harrower [8], but see also [19,7]) have independently proposed polynomial-size integer programming formulations. The LP relaxation of these formulations is quite weak. The use of some valid cuts [8] improves the quality of the bound, but the optimality gap remains large. Brown and Harrower also propose an hybrid model [20], in which a fixed subset of haplotypes are explicitly present, while the rest are implicitly represented by polynomially many variables and constraints. The polynomial/hybrid formulations were successfully used for the solution of problems of similar size as the exponential model. Furthermore, some tests were conducted on larger problems (30 genotypes over up to 75 SNPs), on which the exponential formulation could not be applied successfully due to the IP size.

- **Quadratic, semi-definite programming approaches, of exponential size.** A quadratic formulation, solved by semi-definite programming, was proposed by Kalpakis and Namjoshi [21]. Similarly to the exponential IP, the formulation has a variable for each possible haplotype and cannot be used to tackle instances for which \( \chi(G) \) is too large. The size of the problems solved is comparable to the other methods. Based on a similar formulation, an (exponential-time) approximation algorithm is presented in [22].

- **Combinatorial branch-and-bound approaches.** In [23], Wang and Xu propose a simple combinatorial branch-and-bound approach. The solution is built by enumerating all possible resolutions for each of the genotypes in turn. The lower bound is the number of haplotypes used so far. Since the search space is exponential, and the bound is weak, the method is not able to solve instances of size comparable to the other approaches. Even the solution for 20 genotypes over 20 SNPs can sometimes take an extremely long time to be found.

Generally speaking, all of the above mentioned models run into troubles when trying to solve “large” instances (where the most critical parameter is the number of ambiguous positions per genotype). The exponential models imply the creation of too many variables and/or constraints for obtaining a solution within a reasonable time. The polynomial and combinatorial models, on the other hand, employ quite weak lower bounds, so that closing the gap and terminating the branch-and-bound search is again impossible within a reasonable time. In order to solve large instances of pure parsimony haplotyping one needs to resort to the use of effective heuristics, such as the program Co11haps, by Tininini et al. [24], which can find near-optimal solutions to instances with hundreds of genotypes over a hundred or more SNPs.

### 3.3. Perfect phylogeny

One limitation of parsimony haplotyping is that it does not model haplotypes evolution over time. Haplotypes evolve by mutation and recombination, and an individual can possess two haplotypes such that neither one is possessed by one of his or her parents. The haplotype regions in which no recombination/mutations have ever happened in a
population over time are called *blocks*. For reconstructing haplotype blocks, starting from genotype data of a block in a population, the parsimony model is most appropriate [6]. However, for genotypes spanning several blocks, different models of haplotyping have been considered. One of these is haplotyping for *perfect phylogeny*.

The perfect phylogeny model is used under the hypothesis that no recombination happened, only mutations. We assume that at the beginning there was only one ancestral haplotype, and, in time, new haplotypes were derived from existing haplotypes as follows. If at some point there existed a haplotype $h$ in the population and then a mutation of $h[i]$ happened, passed on to the new generation, a new haplotype $h'$ started to exist, with $h'[j] = h[j]$ for $j \neq i$, and $h'[i] = 1 - h[i]$. We say that $h$ is the “father” of $h'$ in the tree of haplotype evolution. In the *infinite-site coalescent model* [25], once a site has mutated it cannot mutate back to its original state. Hence, the evolution of the haplotypes can be described by a rooted arborescence, in which haplotypes are vertices, and each arc is directed from father to child.

A perfect phylogeny is such an arborescence. Given a set $H$ of haplotypes, and a root haplotype $h^*$, a perfect phylogeny for $H$ is a rooted binary tree $T$ such that:

- The root of $T$ corresponds to $h^*$.
- The leaves of $T$ are in 1-to-1 correspondence with $H$.
- Each position $i \in 1, \ldots, n$ labels at most one edge in $T$.
- For each leaf $h \in H$ and edge $e$ along the path from $h^*$ to $h$, if $e$ is labeled with position $i$, then $h[i] \neq h^*[i]$.

Without loss of generality, it can be assumed that $h^* = 00 \cdots 0$. It can be shown that a perfect phylogeny for $H$ exists if and only if there are no four haplotypes $h^1, \ldots, h^4 \in H$ and two positions $i, j$ such that

$$\{h^a[i]h^a[j], 1 \leq a \leq 4\} = \{00, 01, 10, 11\}.$$  

We can then state the following problem:

**[HAPLOTYPING FOR PERFECT PHYLOGENY]**  
**INSTANCE:** A set $G$ of genotypes.  
**PROBLEM:** Find a set $H$ of haplotypes such that $H$ is resolves $G$ and there is a perfect phylogeny for $H$.

Haplotyping for Perfect Phylogeny was introduced by Gusfield [26], who first showed that the problem is polynomial. The result was obtained by reducing the problem to a well-known graph theory problem, i.e., the *graph realization*, with complexity $O(nm \alpha(n, m))$, where $\alpha$ is the slow-growing inverse Ackerman function. Although the complexity is nearly linear-time, implementing the algorithm for graph realization is very difficult. Much simpler to implement while still very effective algorithms were designed by Bafna et al. [9] and Eskin et al. [10]. These algorithms are based on combinatorial insights in the problem structure and on the analysis of 2-bits patterns implied by the heterozygous sites. The complexity for both algorithms is $O(n^2 m)$. Recently, Ding et al. [27] have described an algorithm for perfect phylogeny haplotyping of complexity $O(nm)$, i.e., linear-time.

### 3.4. Disease association

As discussed in the introduction, haplotypes are very useful for diagnostic and genetic studies, since they are related to the presence/absence of genetic diseases.

In a very simplistic way, a genetic disease can be considered as a malfunctioning of a specific gene. A gene does not function properly when its encoding sequence has been mutated with respect to one of its correct versions. Since each gene is present in two copies (a paternal and maternal copy), it may be the case that either copy is malfunctioning. A genetic disease is called *recessive* if a person shows the symptoms of the disease only when *both* gene copies are malfunctioning. For a recessive disease, one can be a healthy carrier, when one copy is malfunctioning but the other is working properly. Examples of recessive diseases are cystic fibrosis and sickle cell anemia. A genetic disease is called *dominant* if a person shows the symptoms of the disease when *at least one* gene copy is malfunctioning. Examples of dominant diseases are Huntington’s disease and Marfan’s syndrome.

Let us consider the genotypes of a population consisting of healthy and diseased individuals. The haplotypes correspond to the gene sequences. Let us call “good” a haplotype corresponding to a sequence encoding a working gene, and “bad” a haplotype for which the encoded gene is malfunctioning. For a dominant disease, one individual is
diseased if just one of his or her haplotypes is bad, while if the disease is recessive, both haplotypes must be bad to be
diseased.

In the context of haplotyping with respect to a disease, there should exist a coloring into good and bad of the
haplotypes inferred from the genotypes which accounts for the disease. Assuming the disease under study is recessive,
we consider the following problem:

**[HAPLOTYPING FOR DISEASE ASSOCIATION]**

**INSTANCE:** A set \( G_D \) of diseased genotypes. A set \( G_H \) of healthy genotypes.

**PROBLEM:** Find a set \( H \) of haplotypes, partitioned into \( H_G \) (good haplotypes) and \( H_B \) (bad haplotypes) such that

(i) \( H \) resolves \( G_H \cup G_D \).

(ii) \( \forall g \in G_H \), there is a resolution \( g = h' \oplus h'' \) in \( H \), such that \( |\{h', h''\} \cap H_G| \geq 1 \).

(iii) \( \forall g \in G_D \), there is a resolution \( g = h' \oplus h'' \) in \( H \), such that \( |\{h', h''\} \cap H_B| = 2 \).

Note that, by reversing the role of good vs. bad and of healthy vs. diseased, the same formulation can be adopted
to study a dominant disease.

The problem of haplotyping for disease association was introduced by Greenberg et al. in [28] and is studied by
Lancia et al. in [11]. The problem is NP-hard, as shown in [11]. However, real-life data are much simpler to solve than
the artificial instances built in the NP-hardness reduction from satisfiability. In fact, in the same paper it is proved that,
provided that the quite weak constraint of having at least two heterozygous sites per genotype holds, the problem is
polynomially solvable.

Acknowledgments

This research was supported in part with grants PRIN and FIRB. We thank Paola Bertolazzi for many helpful
discussions about haplotyping problems in general. Thanks to IASI, CNR, for their hospitality during the period when
part of this work was done.

References

pp. 1–28.
M. Gribskov, T. Lengauer, I.N. Shindyalov, L.F. Ten Eyck, H. Weissig (Eds.), Proceedings of the Annual International Conference on
Udine, Udine, Italy, 2005 (submitted for publication).
cse–99–6, University of California at Davis, Department of Computer Science, Davis, CA, 1999.