CHAPTER 1

COMBINATORIAL HAPLOTYPING PROBLEMS

Giuseppe Lancia

Dipartimento di Matematica e Informatica, University of Udine, Udine,Italy,

1.1 INTRODUCTION

A few years back, the process of collection of a vast amount of genomic data culminated with the completion of the Human Genome Project [22]. The outcome of the project has brought the confirmation that the genetic makeup of humans (as well as other species) is remarkably well conserved. Generally speaking, the differences at DNA level between any two individuals amount to less than 5% of their genomic sequences. As a consequence, the differences at the phenotype level (i.e., in the way the organisms look) must be caused by small regions of differences in the genomes. The smallest possible region consists of a single nucleotide and is called the *Single Nucleotide Polymorphism* (SNP, pronounced "snip"). SNPs are the predominant form of human genetic variation and their importance can hardly be overestimated; they are used, for example, in medical, drug-design, diagnostic, and forensic applications.

Broadly speaking, a *polymorphism* is a trait common to everybody that can take different values, ranging in a limited set of possibilities, called *alleles* (for simple examples, think of blood group or eye color). In particular, a SNP is a nucleotide site, placed in the middle of a DNA region that is otherwise identical in everybody,

Pattern Recognition in Computational Molecular Biology: Techniques and Approaches,
First Edition. Edited by Mourad Elloumi, Costas S. Iliopoulos, Jason T. L. Wang, and Albert Y. Zomaya.

```
Individual 1, paternal:  taggtccCtatttCccaggcgcCgtatacttcgacgggTctata
Individual 1, maternal:  taggtccGtatttAccaggcgcGgtatacttcgacgggTctata

Individual 2, paternal:  taggtccCtatttAccaggcgcGgtatacttcgacgggTctata
Individual 2, maternal:  taggtccGtatttCccaggcgcGgtatacttcgacgggCctata

Individual 3, paternal:  taggtccCtatttAccaggcgcGgtatacttcgacgggTctata
Individual 3, maternal:  taggtccGtatttAccaggcgcCgtatacttcgacgggCctata
```

Figure 1.1 A chromosome in three individuals. There are four SNPs.

at which we observe a statistically significant variability. A SNP is almost always a polymorphism with only two alleles (out of the four possible). For a nucleotide site to be considered a SNP, it must be the case that the less frequent allele is found in the population with some nonnegligible frequency.

For many living species (e.g., mammals), there exist two sexes and each individual has two parents, one of each sex. The genome of these species is organized in pairs of chromosomes and a single copy of each chromosome pair is inherited from each of the two parents. Organisms in which the genome is organized in pairs of homologous chromosomes are called *diploid* organisms. For a diploid organism, at each 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*.

In Figure 1.1, we illustrate a simplistic example, showing a specific chromosome in three individuals. For each individual, the pair of her chromosome copies is reported. There are four SNPs. The alleles for SNP 1, in this example, are `C` and `G`, while for SNP 4 they are `T` and `C`. Individual 1 is heterozygous for SNPs 1, 2, and 3, and homozygous for SNP 4. Her haplotypes are `CCCT` and `GAGT`. The haplotypes of individual 3 are `CAGT` and `GACC`.

Haplotyping an individual consists in determining her two haplotypes for a given chromosome. With the larger availability in SNP genomic data, recent years have seen the birth of many new computational problems related to haplotyping. These problems are motivated by the fact that it can be difficult and/or very expensive to determine the haplotypes experimentally, so *ad hoc* algorithms have been used to correct data errors or to infer missing data.

In the remainder of this chapter, we will address the haplotyping problem for both a single individual and a set of individuals (a population). In the former case, described in Section 1.2, the input is haplotype data inconsistent with the existence of exactly two parents for an individual. This inconsistency is due to experimental errors and/or missing data. In the latter case, described in Section 1.3, the input data specify for each SNP and each individual in a population whether her parents have contributed the same or different alleles, but do not specify which parent contributed which allele.

1.2 SINGLE INDIVIDUAL HAPLOTYPING

The process of passing from the sequence of nucleotides in a DNA molecule to a string over the DNA alphabet is called *sequencing*. A sequencer is a machine that is fed some DNA and whose output is a string of **A**s, **T**s, **C**s, and **G**s. To each letter, the sequencer attaches a value (confidence level) that essentially represents the probability that the letter has been correctly read.

The main problem with sequencing is that, owing to technological limitations, it is not possible to sequence a long DNA molecule at once. What we can do, however, is to sequence short DNA fragments (also called *reads*), of length of about 1000 nucleotides each, which provide small "windows" to look at the target molecule. To sequence a long DNA molecule, the molecule must first be replicated (*amplified*) by creating many copies of it. These copies are then broken, at random, into several smaller fragments, which are fed to a sequencer that will sequence those of the right size. The amplification phase is necessary so that the reads can have nonempty overlap. From the overlap of two reads, one may infer (through a process called *assembly*) a longer fragment, and so on, until the original DNA sequence has been reconstructed. This is, in essence, the principle of *shotgun sequencing*, a method used by Celera Genomics in the late 1990s to allow the completion of the sequencing of the human genome faster compared with other experimental techniques of the time [62]. In shotgun sequencing, the fragments were read and then assembled back into the original sequence by using sophisticated algorithms and powerful computers.

In Figure 1.2, we illustrate an example in which the two chromosome copies (a) and (b) have been amplified, and then some fragments (denoted by rectangular boxes) have been sequenced. The goal would then be to retrieve (a) and (b), given the set of sequenced fragments. The major difficulty obstructing this goal is that, during the amplification phase, both the paternal and the maternal chromosome copies are amplified together so that the random reads can belong to either the paternal or the maternal original copy. Of course, it is unknown which fragments are paternal and which are maternal, and one of the problems in reconstructing the haplotypes consists in segregating the paternal fragments from the maternal ones. In some cases, there may be pairs of reads, called *mate pairs*, for which it is known that they are either both paternal or both maternal. Mate pairs are due to a particular technique that allows to sequence the ends of a fragment several thousand nucleotides long. Again, one can only read about 1000 nucleotides at each end of the target, but the result is stronger than reading two individual fragments as now it is known that the two reads come from the same chromosome copy. Furthermore, the experiment returns a fairly precise estimate of the distance, expressed in bases, between the two reads of a mate pair.

Even with the best possible technology, sequencing errors are unavoidable. The main errors consist in bases that have been miscalled or skipped altogether. Furthermore, contaminants can be present, that is, DNA coming from organisms other than the one that had to be sequenced. Owing to the experimental errors, the reconstruction

Figure 1.2 Sequence reads and assembly of the two chromosome copies.

of the haplotypes, given the reads coming from sequencing, is not always straightforward and may require correction of the input data. In a general way, the *haplotyping problem for an individual* can then be informally stated as follows:

> *Given inconsistent haplotype data coming from sequencing of an individual's chromosome, find and correct the errors in the data so as to retrieve a consistent pair of haplotypes.*

Depending on what type of errors one is after, there can be many versions of this problem (for surveys on individual haplotyping problems, see, e.g., Schwartz [59] or Zhang et al. [70]). Historically, the formalization of the first haplotyping problems for an individual was given by Lancia et al. [47]. In their work, the *Minimum Fragment Removal* (MFR) and *Minimum SNP Removal* (MSR) problems were introduced, which we briefly discuss here.

Given the fact that at each SNP only two alleles are possible, we can encode them by using a binary alphabet. Hence, in the sequel, the two values that a SNP can take are denoted by `0` and `1`. A haplotype is then simply a string over the alphabet $\{0, 1\}$.

The basic framework for a *single individual haplotyping problem* is as follows. There is a set $\mathcal{S} = \{1, \ldots, n\}$ of SNPs and a set $\mathcal{F} = \{1, \ldots, m\}$ of fragments (i.e., reads coming from sequencing). Each SNP is covered by some of the fragments and can take the values `0` or `1`. Since there is a natural ordering of the SNPs, given by their physical location on the chromosome, the data can be represented by an $m \times n$ matrix M over the alphabet $\{0, 1, -\}$, called the *SNP matrix*. Each column corresponds to a SNP and each row corresponds to a fragment. If fragment i covers a SNP j, then $M[i, j]$ is the value of the allele for SNP j appearing in fragment i. The symbol "`-`" is used to represent a SNP not covered by a fragment (see Figure 1.3a and b for an example of a SNP matrix).

A *gapless* fragment is one covering a set of consecutive SNPs (i.e., the `0`s and `1`s appear consecutively in that row). We say that a fragment has k gaps if it covers $k+1$ blocks of consecutive SNPs (e.g., the fragment `00--101---01` has two gaps). Gaps are mainly due to two reasons: (i) thresholding of low-quality reads (when the sequencer cannot call a SNP `0` or `1` with enough confidence, the SNP is marked with a "`-`"); (ii) mate-pairing in shotgun sequencing (in this case $k = 1$, and the situation is equivalent to having two gapless fragments coming from the same chromosome copy).

Two fragments i and j are said to be *in conflict* if there exists a SNP k such that neither $M[i,k] = -$ nor $M[j,k] = -$ and it is $M[i,k] \neq M[j,k]$. The conflict of two fragments implies that either the fragments do not come from the same chromosome copy or there are errors in the data. Given a SNP matrix M, the *fragment conflict graph* is the graph $G_{\mathcal{F}}(M) = (\mathcal{F}, E_{\mathcal{F}})$ with an edge for each pair of fragments in conflict (see Figure 1.3c). Two SNPs i and j are said to be *in conflict* if there exist two fragments u and v such that the 2×2 submatrix defined by rows u and v and columns i and j contains three `0`s and one `1`, or three `1`s and one `0`. Given a SNP matrix M, the *SNP conflict graph* is the graph $G_S(M) = (S, E_S)$, with an edge for each pair of SNPs in conflict (see Figure 1.3d).

If $G_{\mathcal{F}}(M)$ is a bipartite graph, $\mathcal{F}$ can be segregated into two sets H_1 and H_2 of pairwise compatible fragments. From each set, one can infer one haplotype by fragment overlap. Let h_1 and h_2 be the haplotypes thus obtained. Since h_1 and h_2 correspond to the assembly of the fragments, the single individual haplotyping problems are sometimes also referred to as *fragment assembly haplotyping* problems.

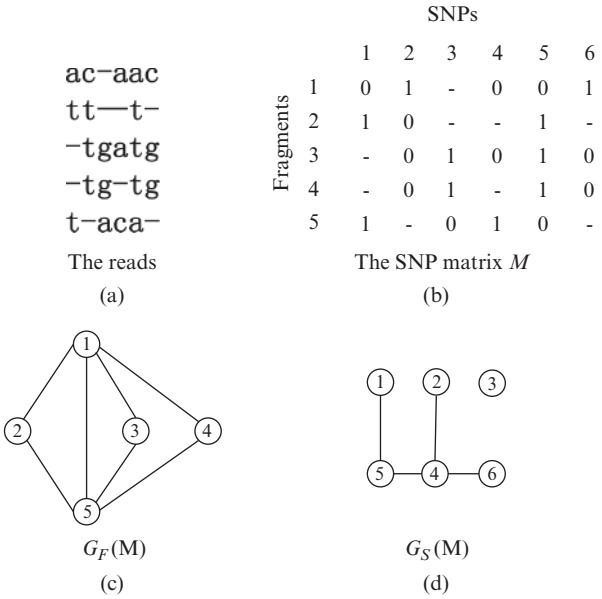

Figure 1.3 SNP matrix and conflict graphs.

We call a SNP matrix M *feasible* if $G_{\mathcal{F}}(M)$ is bipartite and *infeasible* otherwise. Note that a SNP matrix for error-free data must be feasible. Hence, the optimization problems to be defined aim at correcting an infeasible SNP matrix so that it becomes feasible.

The following are the first optimization problems defined with the above goal [47]. They arose at Celera Genomics in the context of sequencing the human genome.

- MFR: Given a SNP matrix, remove the minimum number of fragments (rows) so that the resulting matrix is feasible.
- MSR: Given a SNP matrix, remove the minimum number of SNPs (columns) so that the resulting matrix is feasible.

The first problem is mainly suited for a situation in which, more than sequencing errors, one is worried about the presence of contaminants. The second problem is more suited in the presence of sequencing errors only, when all the fragments are to be retained. These problems were shown to be NP-hard in general [47] so that exact algorithms for their solution are expected to be exponential branch-and-bound procedures. Lippert et al. [53] described a combinatorial branch-and-bound algorithm for MFR. They also described an *Integer Linear Programming* (ILP) formulation of the problem, based on the correspondence between MFR and the *maximum node-induced bipartite subgraph* problem.

While the general case for MFR and MSR is NP-hard, these problems were shown to be polynomial for gapless data [47]. Let us call M a *gapless matrix* if each row of M is a gapless fragment. The main connection between MFR and MSR is given by the following theorem.

Theorem 1.1 *[47] Let M be a gapless SNP matrix. Then, $G_{\mathcal{F}}(M)$ is a bipartite graph if and only if $G_{\mathcal{S}}(M)$ is a stable set.*

Later theoretical improvements extended these results to fragments with gaps of bounded length, giving $O(2^{2l}m^2n + 2^{3l}n^3)$ dynamic programming algorithms for MFR and $O(mn^{2l+2})$ for MSR for instances with gaps of total length l Rizzi et al. [57], Bafna et al. [58]. These algorithms are hardly practical, however, in instances for which the gaps can be rather large. To overcome this problem, Xie and Wang [67] (see also Xie et al. [68]) proposed an algorithm for MFR based on two new parameters: k, the maximum number of SNPs covered by any fragment and c, the maximum number of fragments covering any SNP site (also called *coverage*). Their solution has complexity $O(nc3^c + m \log\ m + mk)$. Since $k \leq n$ and c is generally at most 10, this method should be more suitable for mate-paired data, where l can be quite large.

1.2.1 The Minimum Error Correction Model

Owing to the nature of the sequencing process, most data errors are due to miscalling or missing a base in a read. As a consequence, a particular version of the single

individual haplotyping problem was proposed in Reference [53] and has received great attention because of its practical relevance. This version is called *Minimum Error Correction* (MEC) problem (the problem is also known as *Minimum Letter Flip* (MLF), see Reference [32]):

> *MEC: Given a SNP matrix, change the minimum number of entries (*`0` *into* `1`*, or* `1` *into* `0`*) so that the resulting matrix is feasible.*

Particularly interesting is a weighted version of MEC in which each entry is associated with a nonnegative weight proportional to the confidence level with which the base corresponding to the entry has been called. The solution seeks to flip a set of entries with minimum total weight so as to make the matrix feasible.

The MEC problem was shown to be NP-hard, both for general [53] and for gapless matrices [20]. Many approaches were proposed in the literature for the solution of MEC. For a comparative survey of the various procedures, the reader is referred to Geraci [29] and to Geraci and Pellegrini [30].

One of the first heuristics for MEC was `FastHare` by Panconesi and Suozo [56]. `FastHare` starts by first sorting the fragments according to their leftmost ends and then it reconstructs the two final haplotypes by correcting the sorted fragments in a greedy manner. Being simple, this heuristic is very fast but nevertheless it provides quite accurate solutions in general, especially when the error rate in the data is not too high. For high error-rate data, Genovese et al. proposed `SpeedHap` [28], an effective heuristic procedure organized in phases. For each phase, they perform three tasks: (i) detect likely positions of errors, (ii) allocate fragments to the two partially built final haplotypes, and (iii) decide the final alleles in the two haplotypes via a majority rule on ambiguous SNPs.

`FAHR` (*Fast and Accurate Haplotype Reconstruction* by Wu and Liang [6]) is a somewhat similar greedy heuristic procedure, which builds the final haplotypes by partitioning the fragments at each SNP in a greedy manner. Yet another heuristic for MEC is `HMEC`, proposed by Bayzid et al. [7]. `HMEC` is a steepest–descent local search procedure in which each iteration takes time $O(m^3n)$. Because of its time complexity, `HMEC` may provide results slowly for instances with a large number of fragments.

Zhao et al. [72] considered the *weighted version of MLF* (called WMLF), for which they proposed the use of a dynamic clustering heuristic. Furthermore, they introduced an extended version of the problem to deal with the presence of contaminants. In this version, denoted as Complete Weighted Minimum Letter Flip (CWMLF), one is allowed not only to flip entries of the SNP matrix but also to remove some of the matrix rows. In addition, for CWMLF, they proposed the use of a dynamic clustering algorithm. Wu et al. [66] proposed a heuristic procedure for WMLF, with a performance similar to that of the best previous methods.

Among the various types of approaches to the solution of MEC, there is the use of evolutionary algorithms such as Genetic Algorithms (GA) and Particle Swarm Optimization (PSO). Wang et al. [63] proposed both a branch-and-bound and a GA for MEC. Being exponential, the branch-and-bound is only applicable to instances of very small size, but the GA can be used for large instances (e.g., more than

50 fragments over more than 50 SNPs). A PSO heuristic for MEC was proposed by Kargar et al. [46]. PSO turns out to be fast and suitable for instances with a low error rate in the data.

One of the most successful heuristics for MEC is HapCUT, proposed by Bansal and Bafna [3]. The algorithm makes use of a suitable graph obtained from the fragments, where each SNP corresponds to a node in the graph and two nodes are joined by an edge if there exists a fragment that covers both SNPs. The procedure then tries to minimize the MEC cost of the reconstructed haplotypes by iteratively finding max-cuts in the associated graph.

He et al. [41] proposed a dynamic programming algorithm for MEC with time complexity $O(2^L\ m\ n)$, where L is the maximum length of a fragment. The algorithm can be used for values of L up to about 15, but beyond that, it becomes impractical. For large values of L, the authors suggest to model the problem as a MaxSAT problem, to be solved by a MaxSAT solver. The quality of the solutions obtained through this approach is shown to be quite high, but the solving process is still slow.

Another dynamic programming algorithm, this time parameterized by the maximum coverage c of each SNP site, was proposed by Deng et al. [23]. The complexity of the dynamic program is $O(nc2^c)$, which can be quite high when c is large. For large values of c, the authors propose a heuristic procedure based on the dynamic programming algorithm. Experiments showed that the heuristic returns very accurate solutions on average.

Chen et al. [18] proposed an exact algorithm for MEC based on an ILP formulation of the problem. The solving process is considerably speeded up by a preprocessing phase, in which the input SNP matrix is decomposed into smaller, independent, blocks that are then individually solved by the ILP solver.

1.2.2 Probabilistic Approaches and Alternative Models

Some of the solutions to the assembly haplotyping problems have employed a probabilistic approach, either because of the type of algorithm used or because of the type of model and problem definition.

An example of probabilistic algorithm is HASH (Haplotype Assembly for Single Human), an MCMC (*Markov Chain Monte Carlo*) algorithm proposed by Bansal et al. [4] for assembling haplotype fragments under the MEC objective function. In HASH, the transitions of the Markov chain are generated using min-cut computations on graphs derived from the reads. The method was applied to infer the haplotypes from real data consisting of whole-genome shotgun sequence fragments of a human individual. The results showed the haplotypes inferred by using HASH to be significantly more accurate than the haplotypes estimated by using the best heuristics available at the time.

Another type of Markov chain approach was proposed by Wang et al. [64]. In this approach, the assignment of fragments to haplotypes is regarded as a Markov process in which successive allele pairs are generated on the basis of the value of a small number of preceding SNP sites in the sequence. In order to find the most probable haplotypes for the set of fragments, the authors propose a Viterbi-like dynamic programming procedure.

Chen et al. [19] described a probabilistic model for MEC in which they considered three error parameters, called α_1, α_2, and β. The parameter α_1 is the error rate with which entries of the SNP matrix have been miscalled (i.e., a `0` in place of `1` or a `1` in place of a `0`). The value α_2 is the error rate with which a "`-`" in the SNP matrix appears where, in fact, a SNP is covered by a fragment and therefore an allele `0` or `1` should be present. Finally, β is a measure of the expected dissimilarity between the haplotypes to be reconstructed. The authors gave a linear-time (in the size of M) probabilistic algorithm that reconstructs the correct haplotypes with high probability when the parameters are known. Even for the case when some of the parameters are unknown, they provided a probabilistic algorithm that outputs the correct haplotypes with probability depending on α_1, α_2, and β.

Li et al. [52] proposed a probabilistic model for the haplotype inference problem. In their model, they studied the conditional probability $P(h_1, h_2|M)$ of h_1 and h_2 being the correct haplotypes given that M was the SNP matrix measured. They then pursued the objective of determining the most likely pair of correct haplotypes, that is, a pair $\{h_1^*, h_2^*\}$ maximizing the above conditional probability. In order to solve this problem, they used Gibbs sampling and an *Expectation Maximization* (EM) algorithm.

Among the works proposed for the solution of the MEC model, there are some papers that introduced variants to the original framework in order to deal with some specific data problems. For instance, Zhang et al. [71] proposed a model called *Minimum Conflict Individual Haplotyping* (MCIH), suitable for data sets with particularly high error rate. The problem was shown to be NP-hard and a dynamic programming procedure for its solution was described.

Duitama et al. [25] proposed a model called *Maximum Fragments Cut* (MCF) whose objective is to identify a set of fragments (i.e., rows of the SNP matrix) maximizing a score proportional to their conflict with respect to the remaining rows. This set of fragments can be interpreted as the shore of a cut in a suitable graph, so that the problem can be reduced to a max-cut problem. The authors utilized a local optimization heuristic, called `ReFHap`, for the problem solution.

Xie et al. [69] introduced a model called *Balanced Optimal Partition* (BOP), which generalizes both MEC and MCF. The model is, in a way, a weighted combination of MEC and MCF, and by setting some model parameters to proper values, BOP degenerates into pure MEC or pure MCF. To solve the model, the authors proposed a dynamic programming algorithm called `H-BOP`.

Aguiar and Istrail [1] proposed `HapCompass`, an algorithm for haplotype assembly of densely sequenced human genome data. `HapCompass` operates on a graph where each SNP is a node and the edges are associated with the fragments. The edges are weighted, and the weight of an edge indicates what the best phasing of the alleles for the edge endpoints is, given the fragments in the data set that cover the corresponding SNPs. In this model, the reconstruction of the underlying haplotypes corresponds to a minimum-weight edge-removal problem until a special type of subgraph, called *happy graph* by the authors, is obtained. `HapCompass` is a heuristic with a local reoptimization step. Computational results showed the effectiveness of the procedure and the good quality of its solution also with respect to the MEC objective.

1.3 POPULATION HAPLOTYPING

Haplotype data are particularly sought after in the study of complex diseases (those affected by more than one gene) because they convey precious information about which set of gene alleles are inherited together. These types of polymorphism screenings are usually conducted on a *population*, that is, on sets of related individuals.

We have discussed at length the problem of haplotyping a single individual, which consists in determining her two haplotypes for a given chromosome. In a similar way, *haplotyping a population* consists in haplotyping each individual of the given population. Of course, one way to do this would be to solve a set of single individual haplotyping problems, one for each element of the population, but there might be better ways to proceed. In particular, with the larger availability in SNP data, recent years have seen the birth of a set of new computational problems related to population haplotyping. Most of these problems are motivated by the fact that while it is economically inconvenient to determine the haplotypes by sequencing and assembling, there is a cheap experiment that can determine a less informative and often ambiguous type of data, that is, the *genotypes*, from which the haplotypes can then be retrieved computationally.

A genotype of an individual contains the information about the two (possibly identical) alleles at each SNP, but without specifying their paternal or maternal origin. There may be many possible pairs of haplotypes that are consistent with a given genotype. For example, assume we only know that an individual is heterozygous for the alleles $\{\texttt{C}, \texttt{T}\}$ at SNP 1 and for the alleles $\{\texttt{A}, \texttt{G}\}$ 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 T and G.
(ii) One parent has contributed the alleles C and G and the other the alleles T and A.

Both possibilities are plausible. Associating the alleles with 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 as phasing and haplotyping are in fact the same problem. The two haplotypes that are obtained by phasing the alleles are said to *resolve*, or *explain*, the genotype.

The most general *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.

In other words, the goal is to compute a set H of haplotypes that contains, for each genotype $g \in G$, the two haplotypes h_1 and h_2 obtained by the correct phasing of g. As could be expected, on the basis of only the knowledge of Git, is not easy to describe constraints 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 widely accepted parsimony

principle (inspired by the Occam's razor principle), a good solution may be one that minimizes the number of distinct haplotypes inferred.

Once it has been mathematically modeled, haplotyping gives rise to several nice and challenging combinatorial problems (for surveys on population haplotyping problems, see, e.g., References [11, 38, 39]). These problems have been extensively studied in the last few years. Some of them have been proven NP-hard and solved by exponential-time algorithms, while for others polynomial-time algorithms have been designed.

In this chapter, we address some of the most interesting combinatorial haplotyping models proposed in the literature. Each model and objective function has specific biological motivations, which are discussed in the following sections. While our focus will be on the combinatorial approach to haplotyping problems, it should be remarked that there is also a very important statistical approach to population 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` [60].

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 becomes a binary vector of length n.

A haplotype h is denoted by $h[i]$, the value of its ith component, with $i = 1, \ldots, n$. Given two haplotypes h' and h'', we define a special sum whose result is a vector $g := h' \oplus h''$. The vector g has length n, and its components take 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 Figure 1.4, 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 $\{h', h''\}$ such that $g = h' \oplus h''$. The haplotypes h' and h'' are said to resolve g. A genotype is *ambiguous* if it has more than one possible resolution, that is, 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 can immediately be seen that h is compatible with g if and only if $g[i] = h[i]$ at each position where $g[i] \neq 2$. Two genotypes g and g' are compatible if there exists at least one haplotype compatible with both of them, otherwise, they are *incompatible*. It can immediately be seen that g and g' are compatible if and only if $g[i] = g'[i]$ at each position i where they are both nonambiguous.

As previously discussed, the experiment yielding each genotype is such that, at each SNP, it is known whether an individual is homozygous for allele `0` (in which case, we may set $g[i] = 0$), homozygous for allele `1` (in which case, we may set

Haplotype1,paternal:	0 1 0 1	2 2 2 1	Genotype1
Haplotype1,maternal:	1 0 1 1		
Haplotype2,paternal:	0 0 1 1	2 2 1 2	Genotype2
Haplotype2,maternal:	1 1 1 0		
Haplotype3,paternal:	0 0 1 1	2 0 2 2	Genotype3
Haplotype3,maternal:	1 0 0 0		

Figure 1.4 Haplotypes and corresponding genotypes.

$g[i] = 1$), or heterozygous (in which case, we may set $g[i] = 2$). Therefore, for the haplotyping problems described in this section, 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 a set H of haplotypes such that, for each $g \in G$, there is at least one pair of haplotypes $\{h', h''\} \subseteq H$ 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.

1.3.1 Clark's Rule

The geneticist Clark proposed in Reference [21] a rule (today known as *Clark's rule*) to derive 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.

Note that q and h resolve g. In order to resolve all genotypes of G, Clark suggested the use of successive applications of his inference rule. The procedure requires a "bootstrap" set of haplotypes used to derive new haplotypes at the very beginning. These 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).

The following is the procedure that Clark proposed for haplotyping a population. He supported the validity of the approach by arguments from theoretical population genetics:

Clark's Algorithm: Let $G' \subset G$ be the set of nonambiguous 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 succeeds if $G = \emptyset$ and fails otherwise.

Note that the procedure is nondeterministic as it does not specify how to choose g and 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 nondeterminism in the choice of the genotype g and haplotype 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 Reference [21], a large number of random sorting are used to run the algorithm and the best solution overall is reported. Tests on real and simulated data sets showed that although most of the time the algorithm could resolve all genotypes, often the algorithm failed.

The problem of finding an ordering of application of Clark's inference rule that leaves the fewest number of unresolved genotypes in the end was first defined and studied by Gusfield [33], who proved it is NP-hard and APX-hard (i.e., there is a value $\delta > 1$ for which it is NP-hard even to give a δ-approximation algorithm). As for practical algorithms, Gusfield [34, 35] 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 $N(g) := \{(h, g) : h \ \textit{is compatible with} \ g\}$. Let $N' = \bigcup_{g \in G'} N(g)$ be the subset of haplotypes determined from the set G' of unambiguous genotypes. For each $v = (h, g')$ and $w = (q, g)$ 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, any 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 follows: 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 for every ambiguous genotype g, at most one node in $N(g)$ is reached.

The above graph problem was shown to be NP-hard [34]. For its solution, Gusfield proposed an ILP formulation and noticed that the solution of the LP 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 were ambiguous, over 15 SNPs.

1.3.2 Pure Parsimony

Clark's algorithm tries to reuse existing haplotypes as much as possible and it introduces new haplotypes only when strictly needed. As a result, the procedure usually tends to produce solutions of small size. Note that the maximum size for a set H resolving G is $2|G|$, while the smallest possible size is $\Omega(\sqrt{|G|})$.

Pure Parsimony Haplotyping (PPH) has the explicit objective of minimizing the size of H. This objective has several biological motivations. For one, the number of distinct haplotypes observed in nature is vastly smaller than the number of possible haplotypes. Furthermore, as we all descend from a small number of ancestors, their

haplotypes should be the same we possess today (if it were not for some recombination events and mutations). Finally, pure parsimony follows a general principle according to which, of many possible explanations of an observed phenomenon, one should always favor the simplest.

The PPH problem is NP-hard, as first shown by Hubbel [43]. Lancia et al. [48] showed that, in fact, the problem is APX-hard. This result holds also if each genotype is restricted to possess at most three ambiguous sites. Note that although the problem is APX-hard, there exist constant-ratio approximation algorithms under the restriction that each genotype has at most k ambiguous sites for each constant k [48].

Several algorithmic approaches, many of them employing mathematical programming techniques, have been used in PPH (for a survey of models and approaches for PPH, see Reference [17]). In particular, we recall the following works.

1.3.2.1 Integer Programming Formulations of Exponential Size. Let us denote by $H(G)$ the set of all haplotypes compatible with at least one genotype of G and let $\chi(G) := |H(G)|$. The first ILP formulation for PPH, called *TIP*, was provided by Gusfield [37]. TIP has $\chi(G)$ variables and $O(m2^n)$ constraints. There is a binary variable x_h associated with every $h \in H(G)$ ($x_h = 1$ means that h is included in the solution, whereas $x_h = 0$ means that h is not included). After fixing a total ordering on $H(G)$, denote by $\mathcal{P}$ the set of those ordered pairs (h_1 and h_2) with h_1 and $h_2 \in H(G)$, $h_1 < h_2$. For every $g \in G$, let $\mathcal{P}_g := \{(h_1, h_2) \in \mathcal{P} \mid h_1 \oplus h_2 = g\}$. For every $g \in G$ and $(h_1, h_2) \in \mathcal{P}_g$, there is a binary variable y_{h_1,h_2} used to select an ordered pair (h_1, h_2) to resolve the genotype g. The following is then a valid ILP formulation of the PPH problem [37, 48]:

$$\min \sum_{h \in H(G)} x_h \tag{1.1}$$

subject to

$$\sum_{(h_1,h_2)\in\mathcal{P}_g} y_{h_1,h_2} \geq 1 \qquad \forall g \in G \tag{1.2}$$

$$y_{h_1,h_2} \leq x_{h_1} \qquad \forall (h_1, h_2) \in \bigcup_{g\in G} \mathcal{P}_g \tag{1.3}$$

$$y_{h_1,h_2} \leq x_{h_2} \qquad \forall (h_1, h_2) \in \bigcup_{g\in G} \mathcal{P}_g \tag{1.4}$$

$$x_h \in \{0, 1\}, y_{h_1,h_2} \in \{0, 1\} \qquad \forall h, h_1, h_2 \in H(G) \tag{1.5}$$

Constraints (1.2) impose that each genotype is resolved by at least one pair of haplotypes. Constraints (1.3) and (1.4) make sure that (h_1, h_2) can be used to resolve a genotype only if both h_1 and h_2 are included in the solution.

Gusfield managed to employ some preprocessing rules to get rid of variables that can be proved to be nonessential in the formulation. The resulting model is called RTIP (*Reduced TIP*). Although practically useful, the preprocessing still leaves an exponential model whose size grows quite quickly with respect to the instance size. The experimental results of Reference [37] showed that RTIP can be used to tackle

problems with up to 50 genotypes over 30 SNPs, but there must be relatively small levels of heterozygosity in the input genotypes.

Lancia and Rizzi [50] showed that when each genotype has at most two ambiguous sites, the above ILP is totally unimodular. As a consequence, the ILP solution is always naturally integer and hence the problem can be solved in polynomial time.

Lancia and Serafini [51] proposed a new exponential ILP model for PPH. The model exploits an interpretation of PPH as a particular type of *Set Covering* (SC) based on the following observation: if *H is a set of haplotypes resolving G, then for each genotype g, ambiguous position $i \in A(g)$, and value $a \in \{0, 1\}$, there is a haplotype $h \in H \cap H(g)$ such that $h[i] = a$.*

This condition is a covering condition because it represents the covering constraint for a set cover problem in which the universe is the set of all triples (g, i, a), for $g \in G$, $i \in A(g)$, and $a \in \{0, 1\}$, and each haplotype h represents a subset of the universe, namely, $h \leftrightarrow \{(g, i, a) \mid h \in H \cap H(g), h_i = a\}$. The condition is only necessary, but not sufficient, for H to be a feasible solution of PPH. Consider, for example, the following "diagonal" instance $G = \{1222, 2122, 2212, 2221\}$. The set $\{0111, 1011, 1101, 1110\}$ satisfies the covering condition but does not resolve G.

The SC associated with PPH seeks to minimize the number of haplotypes needed to satisfy all covering conditions for the given set of genotypes G. As we have seen, this SC is in fact a relaxation of PPH. The formulation of the SC model has an exponential number of variables and constraints, and can be solved, as described in Reference [51], by branch-and-cut-and-price, that is, via the generation of variables and constraints at run-time. It is possible that the optimal solution of SC resolves G, in which case it is an optimal PPH solution as well. If not, one can try to obtain a good feasible PPH solution from the optimal cover by adding only a small number of haplotypes. This idea is exploited by an effective heuristic presented in Reference [51]. The computational results show that the SC approach can be orders of magnitude faster than RTIP and can be applied to instances with $n = m = 50$ and $\chi(G)$ up to 10^9. These are among the largest-size instances for which provably optimal solutions have been obtained in the literature.

1.3.2.2 Integer Programming Formulations of Polynomial Size and Hybrid Formulations. Many authors Brown and Harrower [13], Bertolazzi et al. [8], Lancia et al. [48], Halldorsson et al. [40] have independently proposed polynomial-size integer programming formulations. More or less, all these formulations employ variables to represent the bits of the haplotypes in the solution (so there are, at most, $O(mn)$ such variables). The basic idea is that for each genotype $g^i \in G$ one must determine two haplotypes h_1^i and h_2^i such that $h_1^i \oplus h_2^i = g^i$. This implies that, for each position j such that $g^i[j] = 2$, one needs to decide a variable $x_{ij} \in \{0, 1\}$, and then set $h_1^i[j] = x_{ij}$ and $h_2^i[j] = 1 - x_{ij}$. The polynomial formulations for PPH consist in expressing the PPH objective function and constraints in terms of the x variables and possibly a (polynomial-size) set of additional variables.

The LP relaxation of these formulations is generally quite weak. The use of some valid cuts [8, 13] improves the quality of the bound, but the integrality gap between the integer optimum and the LP relaxation remains large. Brown and Harrower [14]

also proposed a hybrid model in which variables for a fixed subset of haplotypes are explicitly present, while the rest of the haplotypes are implicitly represented by polynomially many variables and constraints. These polynomial/hybrid formulations were successfully used for the solution of problems of similar size as those solvable by using TIP. 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 owing to the IP size.

The last polynomial model for PPH was proposed by Catanzaro et al. [16] and is based on the class representatives with smallest index (a technique adopted by Campelo et al. for the vertex color problem [15]). The authors observed that a feasible solution to PPH can be represented by means of a bipartite graph whose two shores correspond to haplotypes and genotypes. Each vertex $g \in G$ has degree 2 and there are two vertices, say h' and h'' adjacent to g such that $g = h' \oplus h''$. The bipartite graph representation of a solution suggests that, in a feasible solution to PPH, the haplotypes induce a family of subsets of genotypes satisfying the following three properties: (i) each subset of genotypes shares one haplotype, (ii) each genotype belongs to exactly two subsets, and (iii) every pair of subsets intersects in at most one genotype. Catanzaro et al. [16] exploited the bipartite graph representation of a solution to PPH to provide a polynomial-size ILP model that turns out to be quite effective and to outperform other polynomial models for the PPH problem.

1.3.2.3 Quadratic, Semidefinite Programming Approaches. A quadratic formulation solved by semidefinite programming was proposed by Kalpakis and Namjoshi [45]. Similarly to the RTIP model, the formulation has a variable for each possible haplotype and hence it cannot be used to tackle instances for which $\chi(G)$ is too large. According to the computational results, the size of the problems solved is comparable to that of RTIP and of the best polynomial ILP models. On the basis of a similar formulation, an (exponential-time) approximation algorithm was presented in Reference [42].

1.3.2.4 Combinatorial branch-and-bound Approaches. Wang and Xu [65] proposed 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.

1.3.2.5 Boolean Optimization. One of the approaches applied to PPH is *Pseudo Boolean Optimization* (PBO). PBO is a technique by which a problem can be modeled via integer linear constraints over a set of boolean variables. The goal is to find an assignment of the variables that satisfies all constraints and minimizes a given objective function. The model is solved by a SAT solver (much in the same way as an ILP model is solved by an ILP solver), which employs solution-searching techniques specific for boolean models.

The PBO models for PPH are mostly based on the following feasibility question: given a tentative cardinality k, does there exist a set H of k haplotypes that resolves G? The feasibility question is expressed in terms of boolean variables similar to the binary variables of the polynomial ILP models. Starting at a lower bound, k is increased until the feasibility question has a positive answer. At that point, H represents the optimal solution.

The first PBO algorithm for PPH was presented by Lynce and Marques-Silva [54]. Later works aimed at breaking the symmetries in the original PBO model, and computing tight lower and upper bounds to be used for pruning the search space Graca et al. [31], Marques-Silva et al. [55]. The SAT approaches showed to be competitive with the best mathematical programming approaches for PPH.

1.3.2.6 Heuristics. In general, all the exact models mentioned so far 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 PPH, one needs then to resort to the use of effective heuristics that can find near-optimal solutions to instances with hundreds of genotypes over hundreds of SNPs with high levels of heterozygosity.

One such heuristic procedure is `CollHaps`, proposed by Tininini et al. [61]. `CollHaps` is based on the representation of the solution as a matrix M' of $2m$ rows (the solving haplotypes, also called *symbolic haplotypes* because they may contain variables), in which some entries are fixed, while others have to be determined. The entries to be determined correspond to heterozygous positions in the input genotypes. To each such entry, `CollHaps` associates a 0–1 variable, which is then set to a fixed value in the course of the procedure. At each step, `CollHaps` tries to maintain as few distinct symbolic haplotypes as possible, which is done by trying to make identical some rows of M' via a greedy setting of the variables. The idea is to eventually end up with as few distinct actual haplotypes as possible. Experimental results have shown that `CollHaps` is a very effective and accurate heuristic for PPH, and ranks among the best available procedures for this problem.

In another heuristic approach for PPH, Di Gaspero and Roli [27] proposed the use of stochastic local search, which they considered to yield a reasonable compromise between the quality of the solutions and running time of the procedure. In their work, Di Gaspero and Roli utilized a family of local search strategies, such as *Best Improvement* (BI), *Stochastic First Improvement* (SFI), *Simulated Annealing* (SA), and *Tabu Search* (TS).

1.3.3 Perfect Phylogeny

One limitation of the PPH model is that it does not take into account the fact that haplotypes may evolve over time. Haplotypes evolve by mutation and recombination,

and an individual can possess two haplotypes such that neither one is possessed by one of her parents. The haplotype regions in which no recombination and/or 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 the most appropriate [37]. However, when genotypes span 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 events happened, but there were only mutations. It is assumed that at the beginning there existed only one ancestral haplotype, and new haplotypes were derived over time 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, which was then 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-sites coalescent model* [44], 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 the haplotypes are the 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 haplotype h^* from which all other haplotypes have evolved, a perfect phylogeny for H is a rooted binary tree T such that the following holds:

1. The root of T corresponds to h^*.
2. The leaves of T are in one-to-one correspondence with H.
3. Each position $i \in 1, \ldots, n$ labels at most one edge in T.
4. 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 $h^* = \texttt{00} \ldots \texttt{0}$. 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

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

The *haplotyping for perfect phylogeny* problem can then be stated as follows: given a set G of genotypes, find a set H of haplotypes such that H resolves G and there is a perfect phylogeny for H.

Haplotyping for perfect phylogeny was introduced by Gusfield [36], who first showed that the problem is polynomial and conjectured the existence of a linear-time algorithm for its solution. To prove that it is polynomial, Gusfield reduced the problem to a well-known graph theory problem, that is, the *graph realization*, with complexity $O(nm\ \alpha(n, m))$, where α 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. [2] and Eskin et al. [26]. These algorithms are based on combinatorial insights in the problem structure and on the analysis of the 2-bit patterns implied by the heterozygous sites. The complexity for both algorithms is $O(n^2\ m)$. Eventually, Ding et al. [24] were able to obtain an algorithm for perfect phylogeny haplotyping of complexity $O(nm)$, that is, a linear-time algorithm. The main idea of the algorithm is to find the maximal subgraphs that are common to all feasible solutions. These subgraphs are represented via an original and effective data structure called a *shadow tree*, which is built and updated in linear time.

Almost at the same time as Ding et al. obtained their result, another linear-time algorithm for perfect phylogeny haplotyping was proposed by Bonizzoni [9], who showed that the problem solution can be obtained via the recognition of special posets of width 2.

The perfect phylogeny model does not take into account events such as back mutations. Bonizzoni et al. [10] considered a variant of the perfect phylogeny model called *Persistent Perfect Phylogeny* (P-PP). In the P-PP model, each SNP site can mutate to a new value and then back to its original value only once. Furthermore, they considered the case of incomplete data, that is, a SNP matrix in which some entries may be missing. They developed an exact algorithm for solving the P-PP problem that is exponential in the number of SNPs and polynomial in the number of individuals.

1.3.4 Disease Association

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

In a very simplistic way, we can define a genetic disease as the 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, it may be the case that either one of the two copies is malfunctioning. A genetic disease is said to be *recessive* if a person shows the symptoms of the disease only when *both* gene copies are malfunctioning (examples of recessive diseases are cystic fibrosis and sickle cell anemia). Note that, for a recessive disease, one can be a healthy carrier, that is, one copy is malfunctioning but the other is working properly. 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 her haplotypes is bad, while if the disease is recessive, both haplotypes must be bad for the individual to be diseased. In the context of haplotyping with respect to a disease, there should exist a coloring of the haplotypes into good and bad that is consistent with the genotypes of healthy and diseased individuals. Assume, for example, that the disease under study is recessive. Then we have

the following problem, called *Haplotyping for Disease Association* (HDA): we are given a set G_D of *diseased* genotypes (i.e., belonging to diseased individuals) and a set G_H of *healthy* genotypes (i.e., belonging to healthy individuals). We want to 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 versus bad and healthy versus diseased, the same formulation can be adopted to study a dominant disease.

The problem of HDA was introduced by Greenberg et al. [32] and was studied by Lancia et al. [49]. The problem is NP-hard, as shown in Reference [49]. 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 has been proved that, provided that the quite weak constraint of having at least two heterozygous sites per genotype holds, HDA is polynomially solvable.

1.3.5 Other Models

We have seen that if an individual is homozygous at a given SNP site, her genotype does not only specify that the individual is homozygous but it also specifies the allele at that particular site. This type of information carries a cost and there also exists a different type of genotype, called *XOR-genotype*, which is cheaper to obtain than a "normal" genotype. The XOR-genotype still distinguishes heterozygous sites from homozygous sites, but it does not identify the homozygous alleles. If we denote by the letter `E` the fact that an individual is heterozygous at a certain SNP site, and by the letter `O` the fact that she is homozygous, an X-OR genotype is then a vector over the alphabet {`E`, `O`}.

When the input data consist of XOR-genotypes rather than normal genotypes, similar population haplotyping problems as those described above can be defined. In particular, Bonizzoni et al. [12] considered the problem of PPH for XOR-genotype data. They gave exact polynomial-time solutions to some restricted cases of the problem, and both a fixed-parameter algorithm and a heuristic for the general case. These results are based on an insight into the combinatorial properties of a graph representation of the solutions.

The perfect phylogeny haplotyping problem for XOR-genotypes was considered by Barzuza et al. [5]. They showed how to resolve XOR-genotypes under the perfect phylogeny model and studied the degrees of freedom in such resolutions. In particular, they showed that when the input XOR-genotypes admit only one possible resolution, the full genotype of at most three individuals would have sufficed in order to determine all haplotypes of the phylogeny.

REFERENCES

1. Aguiar D, Istrail S. Hapcompass: a fast cycle basis algorithm for accurate haplotype assembly of sequence data. J Comput Biol 2012;19(6):577–590.
2. Bafna V, Gusfield D, Lancia G, Yooseph S. Haplotyping as perfect phylogeny: a direct approach. J Comput Biol 2003;10:323–340.
3. Bansal V, Bafna V. HapCUT: an efficient and accurate algorithm for the haplotype assembly problem. Bioinformatics 2008;24:i153–i159.
4. Bansal V, Halpern A, Axelrod N, Bafna V. An MCMC algorithm for haplotype assembly from whole-genome sequence data. Genome Res 2008;18(8):1336–1346.
5. Barzuza T, Beckmann JS, Shamir R, Pe'er I. Computational problems in perfect phylogeny haplotyping: XOR-genotypes and tag SNPs. In: *15th Annual Symposium on Combinatorial Pattern Matching (CPM)*. Volume 3109, Lecture Notes in Computer Science. Springer-Verlag; 2004. p 14–31.
6. Wu J, Liang B. A fast and accurate algorithm for diploid individual haplotype reconstruction. J Bioinform Comput Biol 2013;11(4).
7. Bayzid Sa, Alam M, Mueen A, Rahman S. Hmec: a heuristic algorithm for individual haplotyping with minimum error correction. ISRN Bioinformatics 2013;(Article ID.291741):1–10.
8. Bertolazzi P, Godi A, Labbè M, Tininini L. Solving haplotyping inference parsimony problem using a new basic polynomial formulation. Comput Math Appl 2008;55(5):900–911.
9. Bonizzoni P. A linear-time algorithm for the perfect phylogeny haplotype problem. Algorithmica 2007;48(3):267–285.
10. Bonizzoni P, Braghin C, Dondi R, Trucco G. The binary perfect phylogeny with persistent characters. Theor Comput Sci 2012;454:51–63.
11. Bonizzoni P, Della Vedova G, Dondi R, Li J. The haplotyping problem: an overview of computational models and solutions. J Comput Sci Technol 2004;19(1):1–23.
12. Bonizzoni P, Della Vedova G, Dondi R, Pirola Y, Rizzi R. Pure parsimony XOR haplotyping. IEEE/ACM Trans Comput Biol Bioinform 2010;7:598–610.
13. Brown DG, Harrower IM. A new integer programming formulation for the pure parsimony problem in haplotype analysis. In: *Proceedings of Annual Workshop on Algorithms in Bioinformatics (WABI)*, Lecture Notes in Computer Science. Springer-Verlag; 2004. p 254–265.
14. Brown DG, Harrower IM. A new formulation for haplotype inference by pure parsimony. Technical Report CS-2005-03. Waterloo: University of Waterloo, Department of Computer Science; 2005.
15. Campelo M, Campos V, Correa R. On the asymmetric representatives formulation for the vertex coloring problem. Discrete Appl Math 2008;156(7):1097–1111.
16. Catanzaro D, Godi A, Labbè M. A class representative model for pure parsimony haplotyping. INFORMS J Comput 2010;22:195–209.
17. Catanzaro D, Labbè M. The pure parsimony haplotyping problem: overview and computational advances. Int Trans Oper Res 2009;16:561–584.
18. Chen Z, Deng F, Wang L. Exact algorithms for haplotype assembly from whole-genome sequence data. Bioinformatics 2013;29(16):1938–1945.

19. Chen Z, Fu B, Schweller R, Yang B, Zhao Z, Zhu B. Linear time probabilistic algorithms for the singular haplotype reconstruction problem from SNP fragments. J Comput Biol 2008;15(5):535–546.
20. Cilibrasi R, Iersel LV, Kelk S, Tromp J. On the complexity of several haplotyping problems. In: *Proceedings of Annual Workshop on Algorithms in Bioinformatics (WABI)*. Volume 3692, Lecture Notes in Computer Science. Springer-Verlag; 2005. p 128–139.
21. Clark A. Inference of haplotypes from PCR amplified samples of diploid populations. Mol Biol Evol 1990;7:111–122.
22. Collins FS, Morgan M, Patrinos A. The human genome project: lessons from large-scale biology. Science 2003;300(5617):286–290.
23. Deng F, Cui W, Wang L. A highy accurate heuristic algorithm for the haplotype assembly problem. BMC Genomics 2013;14:1–10.
24. Ding Z, Filkov V, Gusfield D. A linear-time algorithm for the perfect phylogeny haplotyping problem. In: *Proceedings of the Annual International Conference on Computational Molecular Biology (RECOMB)*. New York: ACM Press; 2005.
25. Duitama J, Huebsch T, McEwen G, Suk EK, Hoehe M. Refhap: a reliable and fast algorithm for single individual haplotyping. Proceedings of the 1st ACM International conference on Bioinformatics and Computational Biology, DMTCS'03. New York: ACM; 2010. p 160–169.
26. Eskin E, Halperin E, Karp R. Efficient reconstruction of haplotype structure via perfect phylogeny. J Bioinform Comput Biol 2003;1(1):1–20.
27. Di Gaspero L, Roli A. Stochastic local search for large-scale instances of the haplotype inference problem by pure parsimony. J Algorithms 2008;63:55–69.
28. Genovese L, Geraci F, Pellegrini M. Speedhap: an accurate heuristic for the single individual SNP haplotyping problem with many gaps, high reading error rate and low coverage. IEEE/ACM Trans Comput Biol Bioinform 2008;5(4):492–502.
29. Geraci F. A comparison of several algorithms for the single individual SNP haplotyping reconstruction problem. Bioinformatics 2010;26(18):2217–2225.
30. Geraci F, Pellegrini M. Rehap: an integrated system for the haplotype assembly problem from shotgun sequencing data. In: Fred ALN, Filipe J, Gamboa H, editors. *BIOINFORMATICS 2010 - Proceedings of the 1st International Conference on Bioinformatics*. INSTICC Press: 2010. p 15–25.
31. Graca A, Marques-Silva J, Lynce I, Oliviera AL. Efficient haplotype inference with pseudo-boolean optimization. In: *2nd International Conference on Algebraic Biology (AB)*. Volume 4545, Lecture Notes in Computer Science. Springer-Verlag; 2007. p 125–139.
32. Greenberg H, Hart W, Lancia G. Opportunities for combinatorial optimization in computational biology. INFORMS J Comput 2004;16(3):1–22.
33. Gusfield D. Inference of haplotypes from PCR-amplified samples of diploid populations: complexity and algorithms. Technical Report cse–99–6. Davis (CA): University of California at Davis, Department of Computer Science; 1999.
34. Gusfield D. A practical algorithm for optimal inference of haplotypes from diploid populations. In: Altman R, Bailey TL, Bourne P, Gribskov M, Lengauer T, Shindyalov IN,

Ten Eyck LF, Weissig H, editors. *Proceedings of the Annual International Conference on Intelligent Systems for Molecular Biology (ISMB)*. Menlo Park (CA): AAAI Press; 2000. p 183–189.

35. Gusfield D. Inference of haplotypes from samples of diploid populations: complexity and algorithms. J Comput Biol 2001;8(3):305–324.
36. Gusfield D. Haplotyping as perfect phylogeny: conceptual framework and efficient solutions. In: Myers G, Hannenhalli S, Istrail S, Pevzner P, Waterman M, editors. *Proceedings of the Annual International Conference on Computational Molecular Biology (RECOMB)*. New York: ACM Press; 2002. p 166–175.
37. Gusfield D. Haplotype inference by pure parsimony. In: *Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM)*. Volume 2676, Lecture Notes in Computer Science. Springer-Verlag; 2003. p 144–155.
38. Gusfield D, Orzack SH. Haplotype inference. In: Aluru S, editor. *Handbook of Computational Molecular Biology*. Boca Raton (FL): Champman and Hall/CRC-press; 2005. p 1–28.
39. Halldórsson B, Bafna V, Edwards N, Lippert R, Yooseph S, Istrail S. Combinatorial problems arising in SNP and haplotype analysis. In: *Proceedings of the 4th International Conference on Discrete Mathematics and Theoretical Computer Science, DMTCS'03*. Berlin Heidelberg: Springer-Verlag; 2003. p 26–47.
40. Halldorsson BV, Bafna V, Edwards N, Lippert R, Yooseph S, Istrail S. A survey of computational methods for determining haplotypes. In: *Computational Methods for SNP and Haplotype Inference: DIMACS/RECOMB Satellite Workshop*. Volume 2983, Lecture Notes in Computer Science. Springer-Verlag; 2004. p 26–47.
41. He D, Choi A, Pipatsrisawat K, Darwiche A, Eskin E. Optimal algorithms for haplotype assembly from whole-genome sequence data. Bioinformatics 2010;26(12):i83–i190.
42. Huang YT, Chao KM, Chen T. An approximation algorithm for haplotype inference by maximum parsimony. ACM Symposium on Applied Computing (SAC); 2005. p 146–150.
43. Hubbel E. Unpublished manuscript; 2002.
44. Hudson R. Gene genealogies and the coalescent process. Oxf Surv Evol Biol 1990;7:1–44.
45. Kalpakis K, Namjoshi P. Haplotype phasing using semidefinite programming. 5th IEEE Symposium on Bioinformatics and Bioengineering (BIBE). Minneapolis; 2005. p 145–152.
46. Kargar M, Poormohammadi H, Pirhaji L, Sadeghi M, Pezeshk H, Eslahchi C. Enhanced evolutionary and heuristic algorithms for haplotype reconstruction problem using minimum error correction model. MATCH Commun. Math. Comput. Chem. 2009;62:261–274.
47. Lancia G, Bafna V, Istrail S, Lippert R, Schwartz R. SNPs problems, complexity and algorithms. In: *Proceedings of the Annual European Symposium on Algorithms (ESA)*. Volume 2161, Lecture Notes in Computer Science. Springer-Verlag; 2001. p 182–193.
48. Lancia G, Pinotti C, Rizzi R. Haplotyping populations by pure parsimony: complexity, exact and approximation algorithms. INFORMS J Comput 2004;16(4):17–29.
49. Lancia G, Ravi R, Rizzi R. Haplotyping for disease association: a combinatorial approach. IEEE/ACM Trans Comput Biol Bioinform 2008;5(2):245–251.
50. Lancia G, Rizzi R. A polynomial solution to a special case of the parsimony haplotyping problem. Oper Res Lett 2006;34(3):289–295.

51. Lancia G, Serafini P. A set covering approach with column generation for parsimony haplotyping. INFORMS J Comput 2009;21(1):151–166.
52. Li L, Kim J, Waterman M. Haplotype reconstruction from SNP alignment. J Comput Biol 2004;11:507–518.
53. Lippert R, Schwartz R, Lancia G, Istrail S. Algorithmic strategies for the SNPs haplotype assembly problem. Brief Bioinform 2002;3(1):23–31.
54. Lynce I, Marques-Silva J. SAT in bioinformatics: making the case with haplotype inference. In: *Computational Methods for SNP and Haplotype Inference: DIMACS/RECOMB Satellite Workshop*. Volume 4121, Lecture Notes in Computer Science. Springer-Verlag; 2006. p 136–141.
55. Marques-Silva J, Lynce I, Oliviera AL. *Efficient and Tight Upper Bounds for Haplotype Inference by Pure Parsimony Using Delayed Haplotype Selection*. Volume 4874, Lecture Notes in Artificial Intelligence. Springer-Verlag; 2007. p 621–632.
56. Panconesi A, Sozio M. Fast hare: a fast heuristic for single individual SNP haplotype reconstruction. In: *Proceedings of Annual Workshop on Algorithms in Bioinformatics (WABI)*, Volume 3240, Algorithms in Bioinformatics. Springer-Verlag; 2004. p 266–277.
57. Rizzi R, Bafna V, Istrail S, Lancia G. Practical algorithms and fixed-parameter tractability for the single individual SNP haplotyping problem. In: Guigo R, Gusfield D, editors. *Proceedings of Annual Workshop on Algorithms in Bioinformatics (WABI)*. Volume 2452, Lecture Notes in Computer Science. Springer-Verlag; 2002. p 29–43.
58. Rizzi R, Bafna V, Istrail S, Lancia G. Polynomial and APX-hard cases of the individual haplotyping problem. Theor Comput Sci 2005;335:109–125.
59. Schwartz R. Theory and algorithms for the haplotype assembly problem. Commun Inf Syst 2010;10(1):23–38.
60. Stephens M, Smith N, Donnelly P. A new statistical method for haplotype reconstruction from population data. Am J Hum Genet 2001;68:978–989.
61. Tininini L, Bertolazzi P, Godi A, Lancia G. Collhaps: a heuristic approach to haplotype inference by parsimony. IEEE/ACM Trans Comput Biol Bioinform 2010;7(3):511–523.
62. Venter JC et al. The sequence of the human genome. Science 2001;291:1304–1351.
63. Wang R, Wu L, Li Z, Zhang X. Haplotype reconstruction from SNP fragments by minimum error correction. Bioinformatics 2005;21(10):2456–2462.
64. Wang R, Wu L, Zhang X, Chen L. A markov chain model for haplotype assembly from SNP fragments. Genome Inform 2006;17(2):162–171.
65. Wang L, Xu Y. Haplotype inference by maximum parsimony. Bioinformatics 2003; 19(14):1773–1780.
66. Wu J, Wang J, Chen J. A heuristic algorithm for haplotype reconstruction from aligned weighted SNP fragments. Int J Bioinform Res Appl 2013;9(1):13–24.
67. Xie M, Wang J. An improved (and practical) parametrized algorithm for the individual haplotyping problem MFR with mate pairs. Algorithmica 2008;52:250–266.
68. Xie M, Wang J, Chen J. A model of higher accuracy for the individual haplotyping problem based on weighted SNP fragments and genotype with errors. Bioinformatics 2008;24(13):i105–i113.
69. Xie M, Wang J, Jiang T. A fast and accurate algorithm for single individual haplotyping. BMC Syst Biol 2012;6 Suppl 2:1–10.

70. Zhang XS, Wang RS, Wu LY, Chen L. Models and algorithms for haplotyping problem. Curr Bioinform 2006;1:105–114.
71. Zhang X, Wang R, Wu A, Zhang W. Minimum conflict individual haplotyping from SNP fragments and related genotype. Evol Bioinform Online 2006;2:271–280.
72. Zhao Y, Wu L, Zhang J, Wang R, Zhang X. Haplotype assembly from aligned weighted SNP fragments. Comput Biol Chem 2005;29(4):281–287.