Haplotyping for Disease Association: A Combinatorial Approach

Guiseppe Lancia  
University of Udine

R. Ravi  
Carnegie Mellon University, ravi@cmu.edu

Romeo Rizzi  
University of Udine

Follow this and additional works at: http://repository.cmu.edu/tepper

Part of the Economic Policy Commons, and the Industrial Organization Commons

Published In
IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS, 5, 2.

This Article is brought to you for free and open access by Research Showcase @ CMU. It has been accepted for inclusion in Tepper School of Business by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.
Haplotyping for Disease Association:
A Combinatorial Approach

Giuseppe Lancia, R. Ravi, and Romeo Rizzi

Abstract—We consider a combinatorial problem derived from haplotyping a population with respect to a genetic disease, either recessive or dominant. Given a set of individuals, partitioned into healthy and diseased, and the corresponding sets of genotypes, we want to infer "bad" and "good" haplotypes to account for these genotypes and for the disease. Assume, for example, that the disease is recessive. Then, the resolving haplotypes must consist of bad and good haplotypes so that 1) each genotype belonging to a diseased individual is explained by a pair of bad haplotypes and 2) each genotype belonging to a healthy individual is explained by a pair of haplotypes of which at least one is good. We prove that the associated decision problem is NP-complete. However, we also prove that there is a simple solution, provided that the data satisfy a very weak requirement.

Index Terms—Combinatorial haplotyping, disease association, dominant disease, recessive disease.

1 INTRODUCTION

A single nucleotide polymorphism (SNP, pronounced "snip") is a site of the human genome showing a statistically significant variability within a population. Apart from very rare exceptions, at each SNP, only two nucleotides (out of A, T, C, and G) are observed and they are called the SNP alleles. SNPs are the predominant form of human polymorphism and their importance can hardly be overstated. They are widely used in therapeutic, diagnostic, and forensic applications and a SNP consortium exists with the goal of designing a detailed SNP map for the human genome [13], [10].

Humans are diploid organisms, i.e., their DNA is organized in pairs of chromosomes. For each pair of chromosomes, one chromosome copy is inherited from the father and the other copy is inherited from the mother. For a given SNP, an individual can be either homozygous (i.e., possess the same allele on both chromosomes) or heterozygous (i.e., possess two different alleles). The values of a set of SNPs on a particular chromosome define a haplotype. 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. His or her haplotypes are CCCT and GAGT.

Haplotyping an individual consists of determining his or 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 (see [9] for a survey on haplotyping). These problems are motivated by the fact that it is economically infeasible to determine the haplotypes experimentally. On the other hand, there is a cheap experiment that can determine the (less informative) genotypes. A genotype provides information about the multiplicity of each SNP allele, i.e., for each SNP, a genotype specifies if an individual is heterozygous or homozygous (in the latter case, it also specifies the allele). Since genotypes are much cheaper to obtain than haplotypes, a natural solution for haplotyping has been to define an inference problem to be solved algorithmically: Compute the correct haplotypes from the genotypes.

When retrieving haplotypes from genotypes, there is an inherent ambiguity that comes from heterozygous sites. At each heterozygous site, to retrieve the haplotypes, one has to decide how to distribute the two allele values on the two chromosome copies. Resolving (or explaining) a genotype requires determining the two haplotypes that yield the genotype. Given a set of genotypes, the general (computational) haplotyping problem requires determining a set of haplotypes such that each genotype is explained by two haplotypes. Due to their importance, haplotyping problems have been and are being extensively studied, under many objective functions (each with specific biological motivations). Popular formulations include 1) pure parsimony, which attempts to minimize the total number of distinct haplotypes used to resolve a given set of genotypes. This variant of the problem is APX-hard and several optimization approaches were proposed for its solution [8], [12], [2], [11], [14]. Clark’s rule [3] is a common heuristic toward this end. 2) Perfect phylogeny haplotyping [1], [6], [5] attempts to resolve the genotypes by a set of haplotypes that admit a perfect phylogeny on them.

One of the main reasons why haplotypes are so important and heavily studied is that they are very useful in diagnostic and medical applications 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

Copyright (c) 2008 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a letter to pubs-permissions@ieee.org.
The problem HDA is NP-complete even when restricted to instances in which there are no two individuals, one healthy and one diseased, with the same genotype.

**Theorem 1.** The problem HDA is NP-complete even when restricted to instances in which there are no two individuals, one healthy and one diseased, with the same genotype.

**Theorem 2.** If each genotype has at least two heterozygous sites, then the instance is feasible and the problem is polynomially solvable.

**Theorem 3.** If there exists at least one haplotype that can possibly be used to explain all genotypes (i.e., all genotypes are compatible), then the problem is polynomially solvable.

Let us briefly comment on these results. The negative result of Theorem 1 is, in practice, dominated by the positive result of Theorem 2. In real-life instances, it is expected that each genotype has several heterozygous sites so that each instance should be feasible and polynomially solvable. Moreover, in populations where the most frequent alleles have a much higher frequency than the least frequent ones, it can be shown that genotypes are usually compatible, so Theorem 3 applies.

The existence of a partition of the haplotypes into good and bad is a biologically necessary condition for a solution to be correct and, as such, it has been posed as a combinatorial condition for the haplotyping problem. In our proof of Theorem 2, however, we employ a strictly combinatorial argument that derives a mathematically correct but, most likely, biologically meaningless solution (i.e., bad haplotypes will be such that their alleles, once represented as binary values, sum up to a given remainder modulo 3). Therefore, an important contribution of this work is to show that the partitioning condition alone is too weak to capture the essence of the genetic disease under study and that more constraints must be imposed if the solution is to in fact explain the genetic disease.

### 1.2 Haplotyping as a Combinatorial Problem

Let $n$ be the number of SNPs we consider. Arbitrarily fix a binary encoding of the two alleles for each SNP (e.g., call the least frequent allele “0” and the other “1”). Once the encoding has been fixed, each haplotype is represented by a binary $n$-vector.

For a haplotype $h$, we denote by $h[i]$ the value of its $i$th component. Given two haplotypes $h'$ and $h''$, their sum is a vector $h' \oplus h''$, where the binary operator $\oplus$ is defined componentwise as

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

Fig. 1. The haplotypes of three individuals, with four SNPs.
LANCIA ET AL.: HAPLOTYPING FOR DISEASE ASSOCIATION: A COMBINATORIAL APPROACH

Definition 4 (shorthand notation). which

Definition 3 (compatibility). A haplotype $h$ and $h''$, $g = h' \oplus h''$ is a genotype. The above notation, which defines the sum of two haplotypes, yielding a genotype, is the standard used in all literature on haplotyping problems.

Definition 1 (resolution). For $g$, a genotype, a pair of haplotypes \{h', h''\} such that $g = h' \oplus h''$ is a resolution of $g$. The haplotypes $h'$ and $h''$ are said to resolve $g$. Let $G$ be a set of genotypes and $H'$ and $H''$ be two sets of haplotypes. We say that $(H', H'')$ resolves $G$ if, for every $g \in G$, there exist $h_1 \in H'$ and $h_2 \in H''$ such that $g = h_1 \oplus h_2$. We call such a resolution a resolution in $(H', H'')$ of $g$.

Definition 2 (ambiguity). Let $g$ be a genotype. Each position $i$ such that $g[i] = 2$ is called an ambiguous position. By $n_2(g)$, we denote the number of ambiguous positions of $g$. A genotype is ambiguous if it has more than one resolution, i.e., if $n_2(g) \geq 2$.

In the biological interpretation, genotype entries with a value of 0 or 1 correspond to homozygous SNP sites, while ambiguous positions correspond to heterozygous sites. In Fig. 2, we illustrate a case of three individuals, showing their haplotypes and genotypes.

Definition 3 (compatibility). A haplotype $h$ is compatible with a genotype $g$ if $g[i] = h[i]$ for $g[i] \neq 2$. Two genotypes $g$ and $g'$ are compatible if $g[i] = g'[i]$ whenever $g[i] \neq 2$ and $g'[i] \neq 2$ (i.e., if $g$ and $g'$ share at least one compatible haplotype).

For instance, if $h = 0100$, $g' = 0212$, $g'' = 1222$, and $g''' = 2211$, then $h$ is compatible with $g'$ but not with $g''$ or $g'''$. Moreover, $g'$ and $g''$ are not compatible, while $g'$ and $g'''$ are.

Clearly, a genotype can be resolved only by compatible haplotypes. Given $g$ and $h$ compatible with $g$, it is easy to compute the complement of $h$ with respect to $g$. This is the unique haplotype, denoted by $[g \oplus h]$, for which $h \oplus [g \oplus h] = g$.

Definition 4 (shorthand notation). We denote by 0 a genotype/haplotype of all 0s. For $S \subseteq \{1, \ldots, n\}$, we denote by $1_S$ a genotype/haplotype that is 0 everywhere except at components in $S$, where it is 1. We shortwrite 1, for $1_{\{i\}}$ and $1_{\{i,j\}}$ for $1_{\{i,j\}}$. We denote by $2_S$ a genotype that is 0 everywhere except at components in $S$, where it is 2.

For instance, if $n = 5$, it is $1_3 = 00100$, $1_{1,5} = 10001$, and $2_{1,2,4} = 22020$.

In the problem studied in this paper, the data describe a population of $m$ individuals, of which $m_H$ are healthy and $m_D$ are diseased. An instance consists of two (not necessarily disjoint) sets of genotypes: $G_H$ (the genotypes from healthy individuals) and $G_D$ (the genotypes from diseased individuals). Let $G$ be the multiset obtained by the union of $G_H$ and $G_D$. Each genotype appears in $G$ once or twice (when one healthy and one diseased individual have the same genotype). $G$ can also be viewed as an $m \times n$ matrix with entries in \{0, 1, 2\}, partitioned in submatrices $G_D$ of $m_D$ distinct rows (genotypes) and $G_H$ of $m_H$ distinct rows, with $m = m_D + m_H$.

We consider a recessive disease and we seek two disjoint sets of haplotypes, $H_B$ ("bad" haplotypes) and $H_G$ ("good" haplotypes), that provide a feasible solution to the following problem:

[HAPLOTYPING FOR DISEASE ASSOCIATION (HDA)]

INSTANCE: A set $G_D$ of diseased genotypes. A set $G_H$ of healthy genotypes.

PROBLEM: Find a set of haplotypes, partitioned into $H_G$ and $H_B$ such that

(i) For each $g \in G_D$, there is a resolution of $g$ in $(H_B, H_B)$;

(ii) For each $g \in G_H$, there is a resolution of $g$ in $(H_G, H_G \cup H_B)$.

Notice that there may be infeasible instances of HDA. The smallest such example, for $n = 1$, is to assume that the genotype $g = 2$ is diseased. Then, the only possible haplotypes, i.e., 0 and 1, must both be bad and, hence, there cannot be healthy genotypes. More interesting examples of infeasible instances can be shown, but, basically, they all result from the same basic issue: If some haplotypes are forced to be bad or forced to be good, then the instance may be infeasible. In the following sections, we will prove that this is the only cause of infeasibility and, when each genotype has more than one resolution, the instance is always feasible.

Example. Consider the following instance of HDA:

\[
G_H = \begin{pmatrix} g_1^1 \\ g_2^1 \\ 0222 \\ \end{pmatrix}, \quad G_D = \begin{pmatrix} g_3^1 \\ g_4^1 \\ 2222 \\ \end{pmatrix}, \quad G_D = \begin{pmatrix} g_1^2 \\ g_2^2 \\ 1222 \\ \end{pmatrix}.
\]

Notice that there are two individuals, one healthy and one diseased, with the same genotype. A possible solution is

\[
H_B = \begin{pmatrix} h_1^1 \\ h_2^1 \\ 0100 \\ \end{pmatrix}, \quad H_B = \begin{pmatrix} h_3^1 \\ h_4^1 \\ 1010 \\ \end{pmatrix}, \quad H_B = \begin{pmatrix} h_5^1 \\ h_6^1 \\ 0101 \\ \end{pmatrix}, \quad H_B = \begin{pmatrix} h_7^1 \\ h_8^1 \\ 1101 \\ \end{pmatrix},
\]

with the following resolutions: $g_1^1 = h_1^1 \oplus h_2^1$, $g_2^1 = h_1^1 \oplus h_2^1$, $g_3^1 = h_3^1 \oplus h_4^1$, $g_4^1 = h_3^1 \oplus h_4^1$, and $g_5^1 = h_5^1 \oplus h_6^1$.

2 THE HARDNESS OF HDA

In this section, we prove that HDA is a difficult problem. In particular, we prove the following NP-completeness result:
Theorem 1. Problem HDA is NP-complete even when restricted to instances in which \( G_D \) and \( G_H \) are disjoint.

The proof is based on a reduction from 3SAT, which was shown to be NP-complete in [4].

Let \( \langle U, C \rangle \) be an instance of 3SAT, where \( U = \{ u_1, u_2, \ldots, u_n \} \) is a finite set of Boolean variables and \( C = C_1, C_2, \ldots, C_m \) is a collection of clauses, each containing precisely three literals over \( U \). A literal over \( U \) is either a variable \( u_i \) in \( U \) (positive literal) or its negation \( \overline{u}_i \) (negated literal). In 3SAT, we are asked to find whether there exists a truth assignment \( \Phi : U \mapsto \{ true, false \} \) such that each clause in \( C \) contains at least one literal that evaluates to true under \( \Phi \).

Given the 3SAT-instance \( \langle U, C \rangle \), we construct an instance of HDA as follows:

We introduce one SNP for each literal over \( U \) so that there is a one-to-one correspondence between \( U : \{ u_1, u_2, \ldots, u_n \} \) and the SNPs in the constructed instance of HDA. In our intended interpretation of the reduction, a literal \( \overline{u}_i \in U \) should be read as true if and only if the haplotype \( 1_{\overline{u}_i} \) is in \( H_B \). It remains to be specified how to define \( G_D \) and \( G_H \). First, the genotype 0 is placed in \( G_D \). Next, for each \( i = 1, 2, \ldots, n \) and so as to ensure that at most one of the two literals \( u_i \) and \( \overline{u}_i \) can carry the true value, we place the genotype \( 1_{u_i, \overline{u}_i} \) in \( G_D \) and the genotype \( 2_{u_i, \overline{u}_i} \) in \( G_H \). Finally, for each \( c = 1, 2, \ldots, m \) and assuming that \( u_i, \overline{u}_j, \overline{u}_k \) are the three literals occurring in clause \( C_c \), in order to represent the constraint that at least one of these three literals should evaluate to true, we place the genotype \( 2_{u_i, \overline{u}_j, \overline{u}_k} \) in \( G_D \) and the genotype \( 1_{u_j, \overline{u}_k, \overline{u}_k} \) in \( G_H \).

The description of the reduction is complete. It should be clear that the reduction can be performed in polynomial time.

The following two lemmas conclude our NP-completeness proof:

Lemma 2. Assume that \( \langle U, C \rangle \) admits a satisfying truth assignment. Then, the instance \( \langle G_D, G_H \rangle \) of the HDA constructed above is a Yes instance.

Proof. Let \( \Phi \) be a satisfying truth assignment. First, place 0 in \( H_B \). Next, for each \( i = 1, 2, \ldots, n \), place \( 1_{u_i, \overline{u}_i} \) in \( H_B \). Moreover, if \( \Phi(u_i) = true \), then place \( 1_{u_i} \) in \( H_B \) and \( 1_{\overline{u}_i} \) in \( G_H \); otherwise, do the contrary. For each \( c = 1, 2, \ldots, m \), where \( u_i, \overline{u}_j, \overline{u}_k \) are the three literals occurring in clause \( C_c \), and assuming that \( \Phi(\overline{u}_i) = true \), we can also do possibly renaming the three literals (and remembering that \( \Phi \) is a satisfying truth assignment), place \( 1_{u_i, \overline{u}_j, \overline{u}_k} \) in \( H_B \) and \( 1_{u_j, \overline{u}_k, \overline{u}_k} \) in \( G_H \). The reader is invited to check that \( (H_B, H_B) \) resolves \( G_D \) and \( (H_G, H_G \cup H_B) \) resolves \( G_H \). Moreover, \( H_G \) and \( H_B \) are disjoint.

Lemma 3. Assume that the instance \( \langle G_D, G_H \rangle \) of the HDA constructed above is a Yes instance. Then, the 3SAT-instance \( \langle U, C \rangle \) we started from admits a satisfying truth assignment.

Proof. Let \( H_G \) and \( H_B \) be two disjoint haplotype sets such that \( (H_B, H_B) \) resolves \( G_D \) and \( (H_G, H_G \cup H_B) \) resolves \( G_H \). Clearly, 0 \( \in \) \( H_B \) since 0 \( \in \) \( G_D \). Similarly, since \( 1_{u_i, \overline{u}_i} \) \( \in \) \( G_D \) and \( 1_{u_i} \) \( \in \) \( H_B \) for every \( i = 1, 2, \ldots, n \). Since \( 2_{u_i, \overline{u}_i} \) \( \in \) \( G_H \), both \( 1_{u_i} \) and \( 1_{\overline{u}_i} \) belong to \( H_G \cup H_B \) and at least one of them belongs to \( H_G \). Consider the truth assignment \( \Phi : U \mapsto \{ true, false \} \) defined by \( \Phi(u_i) = true \) if and only if \( 1_{u_i} \in H_G \). We claim that \( \Phi \) is a satisfying truth assignment. Indeed, consider the generic clause \( C_c \) and let \( \overline{u}_i, \overline{u}_j, \) and \( \overline{u}_k \) be the three literals occurring in \( C_c \). Clearly, \( 1_{u_i, \overline{u}_j, \overline{u}_k} \in H_G \) since \( 1_{u_i} \) \( \in \) \( H_G \). Since \( 2_{u_i, \overline{u}_j, \overline{u}_k} \) \( \in \) \( G_H \), it must hold that, for at least one of these three literals, say, \( \overline{u}_i \), we have \( 1_{u_i, \overline{u}_j, \overline{u}_k} \in H_B \). Without loss of generality, let \( 1_{u_i} \) \( \in \) \( H_B \) and, from our assignment rule above, it follows that \( \Phi(\overline{u}_i) = false \). We have thus argued that, in each clause \( C_c \), there is at least one literal \( \overline{u}_i \) such that \( \Phi(\overline{u}_i) = true \), that is, \( \Phi \) is a satisfying truth assignment, as claimed. □

3 POLYNOMIALLY SOLVABLE INSTANCES OF HDA

3.1 All Genotypes with At Least Two Heterozygous Sites

The following theorem says that all instances are feasible, except when some genotypes have only one resolution.

Theorem 4. Assume that, for each genotype \( g \in G \), \( n_2(g) \geq 2 \). Then, the instance is feasible and a feasible solution can be readily obtained.

Proof. Let us divide the set of all possible haplotypes into three classes, \( H_0 \), \( H_1 \), and \( H_2 \) depending on the remainder in the sum of the bits divided by three. More formally,

\[
H_i = \left\{ h : \left( \sum_{i=1}^{n} h_i \right) \mod 3 = i \right\}, \quad \text{for} \ i = 0, 1, 2. \tag{1}
\]

Similarly, divide all of the genotypes into three classes and define \( G_0 \), \( G_1 \), and \( G_2 \) as follows:

\[
G_i = \{ g \in G : n_2(g) \mod 3 = i \}. \tag{2}
\]

We first describe how to resolve genotypes that have \( 2 \leq n_2(g) \leq 4 \), and then, we show how the same type of solution can be applied to all genotypes. The main idea is the following: We are going to make all haplotypes in \( H_0 \) good. We then need to show that, for each healthy genotype, there is a resolution that uses at least a haplotype in \( H_0 \) and, for each diseased genotype, there is a resolution that does not use a haplotype in \( H_0 \).

For a genotype \( g \), we define \( h(g) \) as the haplotype obtained by replacing each 2 in \( g \) with a 0. Given a \( g \) with \( 2 \leq n_2(g) \leq 4 \), each resolution will consist of two haplotypes \( h' \) and \( h'' \) that “contain” \( h(g) \) and of which one has \( k \) more 1s and the other has \( n_2(g) - k \) more 1s than \( h(g) \) does. Depending on \( g \), all of the possibilities for the pair \( k, n_2(g) - k \) are described in Table 1.

Notice that there is always a resolution in which we can add 0, 1, or 2 (modulo 3) to the parity of \( h(g) \) and, hence, we can make the resulting haplotype belong to any class \( H_i \) we want. Furthermore, notice that there is always a resolution that skips, in both haplotypes, adding 0, 1, or 2 to the parity of \( h(g) \) and, hence, we can make the resulting haplotypes not belong to any class \( H_i \) we want.
Assume that all genotypes in \( G \) are mutually compatible. Then, the problem is polynomially solvable.

**Theorem 5.** Assume that all genotypes in \( G \) are mutually compatible. Then, the problem is polynomially solvable.
We should then start any attempt to determine the feasibility of a general instance in which there are nonambiguous genotypes with the following cascade of implications:

1. Let $B$ and $G$ be the set of forced bad and good haplotypes. Initially, $B$ consists of all haplotypes compatible with genotypes in $G_D$ that have $<2$ ambiguous sites. Similarly, $G$ consists of all haplotypes compatible with genotypes in $G_H$ that have no ambiguous sites. Remove from $G_H$ and $G_D$ all genotypes used to derive $B$ and $G$.

2. If $B \cap G \neq \emptyset$, then stop; the problem is infeasible. Otherwise, loop through 3-6 until, for a complete iteration, neither $B$ nor $G$ changes.

3. Let $G'$ be the subset of genotypes in $G_H$ that can be obtained as $h \oplus h'$, with $h, h' \in B$. If, for any $g \in G'$, all of the compatible haplotypes are in $B$, stop; the problem is infeasible.

4. For all $g \in G'$ such that there exists only one haplotype $h$ not in $B$ compatible with $g$, put $h$ in $G$ and remove $g$ from $G_H$.

5. Let $G'$ be the subset of genotypes in $G_D$ that can be obtained as $h \oplus h'$, with $h, h' \in G$. If, for any $g \in G'$, all of the possible resolutions use a haplotype that is in $G$, stop; the problem is infeasible.

6. For all $g \in G'$ such that there exists exactly one resolution $h \oplus h'$ in which both $h$ and $h'$ are not in $G$, put $h$ and $h'$ in $B$ and remove $g$ from $G_D$.

At the end of the cascade, either the problem has been declared infeasible or there are two sets $B$ and $G$ of bad and good haplotypes that must be fixed and all of the unsolved genotypes have a degree of freedom.

As a corollary of Theorem 4, we have that the problem is feasible as long as the haplotypes in $B$ fall in suitable parity classes (e.g., the bad fixed haplotypes skip a certain class modulo 3, which we can then declare good and use to resolve the remaining healthy genotypes).

4 CONCLUSIONS

Given a population affected by a genetic disease, it is expected that all haplotypes can be partitioned into working and faulty haplotypes (what we called “good” and “bad” in this paper). This is a biologically necessary condition for a solution to be correct and, as such, it has been posed as a combinatorial condition for the haplotyping problem. We have shown that it is NP-complete to satisfy this condition. Most importantly, however, we have shown that the partitioning condition alone is too weak to capture the essence of the genetic disease under study (it is hardly believable that a genetic illness is due to the sum of alleles being a multiple of 3...). A contribution of our paper is that it proves that more constraints must be imposed in order for the solution to explain the genetic disease. For instance, it is usually required that a set of haplotypes derived from a set of genotypes either fits a perfect phylogeny or is of the smallest possible size, or both. Hence, we could require either of these conditions as an extra condition for HDA, thereby perhaps forbidding a mathematically correct (but biologically meaningless) solution such as that of Theorem 4. Another possible approach is the following: Assume that an HDA instance is feasible. We have seen that this may be due to the large degree of freedom given by heterozygous sites, which we used to give the “coloring” solution of Theorem 4.
the critical SNPs that are really associated to the disease may be just a small subset of all SNPs. A natural question then is what is the minimum set of SNPs for which the solution is still feasible? We are then led to the following optimization problem: Remove the largest number of SNPs so that the instance left is still feasible. We leave these variants of the HDA problem as directions for future research.

ACKNOWLEDGMENTS

This research was done while Giuseppe Lancia was visiting Carnegie Mellon University. This research was supported by US National Science Foundation (NSF) ITR Grant CCR-0122581 (the ALADDIN project), by NSF Grant CCF-043075, and by MIUR Grant P.R.I.N. “Algoritmi di ottimizzazione per l’analisi comparativa di dati genomici di grandi dimensioni.”

REFERENCES


Giuseppe Lancia received the MS and PhD degrees in algorithms, combinatorics, and optimization from Carnegie Mellon University. He is an associate professor of operations research at the University of Udine. From 1999 to 2001, he was a visiting scientist at Sandia National Laboratories and then at Celera Genomics, where he worked on the completion of the first draft sequencing of the human genome. His research focuses on combinatorial optimization problems arising in computational molecular biology. He has been an associate editor for special issues of INFORMS Journal on Computing and BMC Bioinformatics and has served as a program committee member for several international computational biology conferences.

R. Ravi received the PhD degree in computer science from Brown University. He is the Carnegie Bosch professor of operations research and computer science at Carnegie Mellon University. His research interests include combinatorial optimization, approximation algorithms, and computational molecular biology. He serves on the editorial board of Management Science, ACM Transactions on Algorithms, and Operations Research and has been a program committee member for several computer science and operations research international conferences. He has received the US National Science Foundation CAREER Award and the Sigma Xi Outstanding Graduate Research Prize.

Romeo Rizzi received the Laurea degree in electronic engineering from the Politecnico di Milano in 1991 and the PhD degree in computational mathematics and informatics from the University of Padova, Italy, in 1997. Afterward, he held postdoctoral and other temporary positions at research centers like CWI (Amsterdam), BRICS (Aarhus, Denmark), and IRST (Trento, Italy). In 2001, he became an assistant professor at the University of Trento. Since 2005, he has been an associate professor at the University of Udine. He has a background in operations research and his main interests are in combinatorial optimization and algorithms. He is an area editor of 4OR and acts as a reviewer for the American Mathematical Society. He has published more than 50 articles in a broad range of scientific journals in the areas of discrete mathematics, combinatorics, and algorithms. Since 2004, he has been a trainer of the Italian team for the IOI. His biography was included in the 2007 edition of Who’s Who in the World.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.