Haplotype Inference with Boolean Constraint Solving: An Overview

Inês Lynce
IST/INESC-ID
TU Lisbon
Portugal
ines@sat.inesc-id.pt

Ana Graça
IST/INESC-ID
TU Lisbon
Portugal
assg@sat.inesc-id.pt

João Marques-Silva
School of ECS
U of Southampton
United Kingdom
jpms@ecs.soton.ac.uk

Arlindo L. Oliveira
IST/INESC-ID
TU Lisbon
Portugal
aml@inesc-id.pt

Abstract

Boolean satisfiability (SAT) finds a wide range of practical applications, including Artificial Intelligence and, more recently, Bioinformatics. Although encoding some combinatorial problems using Boolean logic may not be the most intuitive solution, the efficiency of state-of-the-art SAT solvers often makes it worthwhile to consider encoding a problem to SAT. One representative application of SAT in Bioinformatics is haplotype inference. The problem of haplotype inference under the assumption of pure parsimony consists in finding the smallest number of haplotypes that explains a given set of genotypes. The original formulations for solving the problem of Haplotype Inference by Pure Parsimony (HIPP) were based on Integer Linear Programming. More recently, solutions based on SAT have been shown to be remarkably more efficient. This paper provides an overview of SAT-based approaches for solving the HIPP problem and identifies current research directions.

1. Introduction

Haplotype inference is nowadays one of the most challenging problems in human genetics. The identification of haplotypes, that contain the genetic data inherited from each parent, may bring new insights to the genetic predisposition to disease, as well as to response to drugs. Considering the huge amount of data to deal with and the intrinsic complexity of the problem, haplotype inference also poses a number of computational challenges. This is true regardless of the approach followed to infer the haplotypes.

One of the existing approaches for solving the haplotype inference problem is pure parsimony [7]. Given that a solution has to be parsimonious, the original problem becomes a minimization problem. The first tools developed for solving the Haplotype Inference by Pure Parsimony (HIPP) problem were based on Integer Linear Programming (ILP) and include exponential and polynomial size models [7, 1, 2]. More recently, new tools based on Boolean Satisfiability (SAT) and Pseudo Boolean Optimization (PBO) have been developed [11, 5]. The SAT/PBO-based tools use polynomial-size models and additional techniques for further reducing the size of the model. While the former HIPP tools could only solve small-size illustrative problem instances, the latter tools are remarkably more efficient, and capable of solving much larger and harder problem instances.

The recent algorithmic developments for solving the HIPP problem have made the pure parsimony approach competitive to the point that it can be considered an effective alternative to other more standard approaches. Not surprisingly, there is no clear best approach, i.e. different problem instances are solved differently by different approaches. However, there is still a comprehensive evaluation to be done in order to characterize the positive/negative aspects of each approach.

This paper describes the state of the art in haplotype inference by pure parsimony. The next section gives the preliminaries, followed by an introduction to haplotype inference. Section 4 describes the techniques that can be applied in general to solve the HIPP problem. Afterwards, models based on ILP and SAT/PBO are described. Practical evidence shows the performance of different tools and the accuracy of bounding techniques. Finally, we point out future research directions.

2. Preliminaries

DNA (deoxyribonucleic acid) refers to any of the nucleic acids (adenine (A), cytosine (C), guanine (G) and thymine (T)) that are the basis of the genetic information for living organisms. DNA may also refer to the double strand molecule where each strand contains a sequence of bases, and each base contains one nucleic acid. From one strand it is possible to determine the bases of the other strand because A only pairs with T and C with G. For example, given the sequence CCTAAG, the corresponding bases in

the other strand must be GGATTC.

The genetic information is organized in DNA segments called genes. Each gene encodes a specific function. Within cells, DNA is organized into structures called chromosomes, for which a set of genes can be distinguished. In diploid organisms, chromosomes are organized in pairs, due to the inheritance coming from each parent. (In what follows we will only consider diploid organisms.)

DNA contains relevant hereditary information. This information, however, is mostly the same for all human beings. There are some exceptions though, including the ones derived from mutations at specific sites of the DNA strand. These mutations are called Single Nucleotide Polymorphisms (SNPs), assuming they occur in at least 1% of the population. For example, a SNP occurs if the sequence CCTAAG is modified to CCTGAG. The site for which the mutation occurred contains two possible values (called alleles): A and G. The analysis of SNPs is relevant to the extent that they can be related with genetic diseases as well as to patients response to drugs.

3. Haplotype Inference

A haplotype is a sequence of SNPs that are known to be statistically associated. As a consequence of this association, it is often possible to identify just a few SNPs (called tag SNPs) within a haplotype that unambiguously identify the remaining SNPs.

Due to technical limitations, it is not possible to directly obtain haplotypes, which would make possible to distinguish between the SNPs inherited from each of the parents. Instead, genotypes representing the conflated data of the two parents are obtained. SNPs in genotypes are traditionally represented as AA, Aa or aa, where 'A' stands for the original base and 'a' for the mutant.

If both parents have the same DNA base at a given site (and so it is either AA or aa), it is called an homozygous site, and it is straightforward to infer the value of the haplotypes at that site. However, for a heterozygous site (Aa) each haplotype at that site has a different value: one has value A and the other has value a. Hence, for a sequence of SNPs representing a genotype with n heterozygous positions, there are 2^{n-1} possible pairs of haplotypes.

Without lack of generality, in what follows we will assume that genotypes are represented by a sequence of elements that may assume values 0, 1 or 2. The values 0 and 1 represent homozygous sites whereas value 2 represents heterozygous sites. Haplotypes are therefore represented by a sequence of values 0 and 1.

Definition 1 *(Haplotype Inference) Given a set $\mathcal{G}$ of n genotypes, each one represented by a string of size m over the alphabet $\{0,1,2\}$, the haplotype inference problem consists in finding a set $\mathcal{H}$ of haplotypes, each one represented by a string of size m over the alphabet $\{0,1\}$, such that each genotype is* explained *by a pair of haplotypes. A genotype $g_i \in \mathcal{G}$ is explained by a pair of haplotypes $h_j, h_k \in \mathcal{H}$, i.e. $g_i = h_j \otimes h_k$, iff:*

- *if $g_{il} = 0$ then $h_{jl} = h_{kl} = 0$,*
- *if $g_{il} = 1$ then $h_{jl} = h_{kl} = 1$,*
- *if $g_{il} = 2$ then $h_{jl} \neq h_{kl}$,*

where l refers to the l^{th} character of the string.

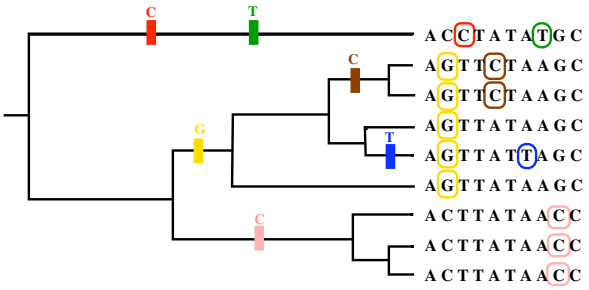

Figure 1. Mutations within a population

Example 1 *(Haplotype Inference) Consider the following set of genotypes $\mathcal{G} = \{g_1, g_2, g_3, g_4\} = \{011, 021, 122, 212\}$. One solution to the haplotype inference problem for $\mathcal{G}$ is the set of haplotypes $\mathcal{H} = \{h_1, h_2, h_3, h_4, h_5\} = \{011, 001, 111, 100, 110\}$, where $g_1 = h_1 \otimes h_1$, $g_2 = h_1 \otimes h_2$, $g_3 = h_3 \otimes h_4$ and $g_4 = h_1 \otimes h_5$.*

There are different approaches for choosing, between the candidate haplotypes, which ones are the most adequate to explain a genotype. This is usually done considering not only one genotype but rather a set of genotypes from individuals of the same population. With such data, it is possible to take into account the coalescent model [9]. This model states that there is a unique ancestor for all individuals of the same population. Hence, the individuals can be grouped accordingly to the mutations they have been affected by. Figure 1 illustrates the effect of mutations within a population, as well as the similarities between individuals.

The coalescent model has inspired statistical approaches that are behind the most well-known tools, which are commonly used by biologists. An alternative approach is pure parsimony, for which the goal is to minimize the number of haplotypes required to explain a given set of genotypes [7]. Although not directly, this approach may also be related with the coalescent model.

Definition 2 *(Haplotype Inference by Pure Parsimony) Given a set of genotypes, a solution to the haplotype inference by pure parsimony (HIPP) problem requires the explaining set of haplotypes to have minimum size.*

Example 2 *(Haplotype Inference by Pure Parsimony) Consider again the set of genotypes* $\mathcal{G} = \{g_1, g_2, g_3, g_4\} = \{011, 021, 122, 212\}$. *A solution to the HIPP problem requires only 4 haplotypes:* $\mathcal{H} = \{h_1, h_2, h_3, h_4\} = \{011, 001, 110, 101\}$, *where* $g_1 = h_1 \otimes h_1$, $g_2 = h_1 \otimes h_2$, $g_3 = h_3 \otimes h_4$ *and* $g_4 = h_1 \otimes h_3$.

The HIPP problem is NP-hard [10].

4. Standard Techniques for Solving HIPP

When solving the HIPP problem, there are quite a few techniques that may be applied during preprocessing. These techniques are inexpensive and empirical evidence shows that they can significantly improve the performance of the HIPP solvers.

4.1. Simplifying the Problem Instances

A key approach for simplifying the haplotype inference problem instances consists in reducing the size of the instance [2].

The set of genotypes given to HIPP solvers contains genotypes from individuals that belong to the same population. Not surprisingly, these sets often contain repeated genotypes, even though each of them refers to different individuals. Clearly, for each subset of repeated genotypes only one of them has to be kept. After a solution to the simplified problem has been found, it is straightforward to find a solution to the original problem.

Other techniques for reducing the size of a problem instance entail removing sites of the genotypes. Consider a set of genotypes, each with the same number of sites. If there are two sites with exactly the same value for each genotype, then one of them can be removed. Furthermore, the same procedure can be applied to symmetric sites. Two sites are said to be symmetric if for each genotype the two sites are either homozygous with value 0(1) and value 1(0) or heterozygous (both with value 2). Again, after a solution to the simplified problem has been found, it is straightforward to find a solution to the original problem.

Example 3 *(Simplification Techniques) Consider the set of genotypes* $\mathcal{G} = \{10111, 10121, 21022, 10111, 12211\}$. *By removing duplicated genotypes, the forth genotype is removed and the set becomes* $\mathcal{G}' = \{10111, 10121, 21022, 12211\}$. *This set is further reduced by removing duplicated sites, which implies removing the fifth site for being equal to the first site, thus becoming* $\mathcal{G}'' = \{1011, 1012, 2102, 1221\}$. *Finally, we may remove the third site for being symmetric to the second site, thus getting the simplified set* $\mathcal{G}''' = \{101, 102, 212, 121\}$.

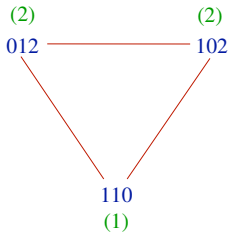

Figure 2. Clique-based lower bound

4.2. Computing Lower Bounds

The techniques for computing lower bounds rely on information regarding incompatible genotypes: two genotypes are *incompatible* if they are both homozygous at the same site but with different values.

A lower bound can be computed from a maximal clique [11]. Clearly, for two incompatible genotypes, g_i and g_l, the haplotypes that explain g_i must be distinct from the haplotypes that explain g_l. Given the incompatibility relation we can create an incompatibility graph I, where each vertex is a genotype, and two vertexes are connected with an edge if they are incompatible. Suppose I has a clique of size k. Then the number of required haplotypes is at least $2k - \sigma$, where σ is the number of genotypes in the clique which do not have heterozygous sites.

Since this problem is NP-hard, we use the size of a clique in the incompatibility graph, computed using a simple greedy heuristic. The genotype with the highest number of incompatible genotypes is first selected. At each step, the genotype selected is one that is still incompatible with all the already selected genotypes, and preference is given to the haplotype with the highest number of incompatible genotypes.

Example 4 *(Lower Bounds) Consider the following set of genotypes:* $\{110, 012, 102\}$. *The three genotypes are incompatible, which is represented in the incompatibility graph in Figure 2, along with each genotype contribution to the lower bound. Hence, the number of required haplotypes is at least 5 (twice the clique size less the number of genotypes with no heterozygous sites).*

In addition, the analysis of the structure of the genotypes allows the lower bound to be further increased, by identifying heterozygous sites which require at least one additional haplotype given a set of previously chosen genotypes [12]. The procedure starts from the clique-based lower bound and grows the lower bound by searching for heterozygous sites among genotypes not yet considered for lower bounding purposes. For each genotype g_i not in the clique, if the genotype has a heterozygous site and all compatible genotypes have the same value at that site (either 0 or 1), then

g_i is guaranteed to require one additional haplotype to be explained. Hence the lower bound can be increased by 1.

Another improvement to the lower bound consists in identifying genotypes with triples of heterozygous sites, among the genotypes not used in the clique lower bound.

Example 5 *(Improved Lower Bounds) Consider the following set of genotypes:* $\{200, 020, 002, 222\}$. *Given that there are no two incompatible genotypes, the clique-based lower bound would give a lower bound of 2 corresponding to a unique vertex (e.g. with the first genotype). The analysis of the structure of the remaining genotypes requires one additional haplotype for the second and the third genotype, thus increasing the lower bound to 4 haplotypes. This lower bound can be further improved by analyzing the fourth haplotype 222. Any of the haplotypes already included in the lower bound requires* at least *two positions with value 0. But the pair of haplotypes explaining 222 will require one haplotype with* at most *one position with value 0. Hence, the lower bound can be increased by 1 to 5.*

4.3. Computing Upper Bounds

Clark's method is a well-known algorithm to solve the haplotype inference problem [3]. This method starts by identifying genotypes with zero or one heterozygous sites, which have only one possible explanation. Then, the method attempts to explain the remaining genotypes with at least one of the haplotypes already identified. This may eventually require the inference of new haplotypes which will be added to the set of haplotypes. The key point to note is that there are many ways to extend the set of haplotypes, since for genotypes with more than one heterozygous site there are a few possible explanations.

Clark's method may be used to compute an upper bound to the HIPP problem. However, this method is often too greedy. An alternative algorithm (called Delayed Selection (DS) [16]) addresses the main drawback of Clark's method. The DS algorithm maintains two sets of haplotypes: the *selected* haplotypes, which represent haplotypes which have been chosen to be included in the target solution, and the *candidate* haplotypes, which represent haplotypes which can explain one or more genotypes not yet explained by a pair of selected haplotypes.

The initial set of selected haplotypes corresponds to all haplotypes which are required to explain the genotypes with no more than one heterozygous sites, i.e. genotypes which are explained with either one or exactly two haplotypes. At each step, the DS algorithm chooses the candidate haplotype h_c which can explain the largest number of genotypes. The chosen haplotype h_c is then used to identify additional candidate haplotypes. Moreover, h_c is added to the set of selected haplotypes, and all genotypes which can be explained by a pair of selected haplotypes are removed from the set of unexplained genotypes. The algorithm terminates when all genotypes have been explained.

Each time the set of candidate haplotypes becomes empty, and there are still more genotypes to explain, a new candidate haplotype is generated. The new haplotype is selected greedily as the haplotype which can explain the largest number of genotypes not yet explained. Given that the proposed organization allows selecting haplotypes which will not be used in the final solution, the last step of the algorithm is to remove from the set of selected haplotypes all haplotypes which are not used for explaining any genotypes.

5. Solving HIPP with ILP

The first solutions for solving the HIPP problem were ILP models, solved with dedicated solvers [7, 8, 1, 2]. These models are briefly reviewed below.

5.1. Exponential-Size ILP Models

The original ILP models, *TIP* and *RTIP*, have linear space complexity on the number of candidate haplotypes [7] and so both are exponential on the number of given genotypes in the worst-case. For each genotype g_i, all r candidate pairs of haplotypes that can explain g_i are enumerated. For example, given genotype 02122, the candidate pairs of haplotypes for explaining it are: (00100,01111), (01100,00111), (00110,01101) and (00101,01110). In the general case, each genotype having k heterozygous sites is explained by 2^{k-1} pairs of haplotypes. Hence, the space complexity is $\mathcal{O}(2^m)$ where m is the number of sites, which represents the maximum number of heterozygous sites per genotype. A Boolean variable $y_{i,u}$ is associated with each pair u of haplotypes that can explain a given genotype g_i; its value is 1 if this pair of haplotypes is used for explaining g_i or 0 otherwise. A cardinality constraint, $\sum_r y_{i,u} = 1$, requires that exactly one pair of haplotypes must be used for explaining each genotype, among all pairs that can explain the genotype. Each candidate haplotype is associated with a dedicated variable x_v, such that $x_v = 1$ if the haplotype is *used*. The utilization of a specific pair of haplotypes for explaining a genotype (i.e. $y_{i,u} = 1$) implies the respective x_v variable, $y_{i,u} \rightarrow x_v$, for each haplotype in the pair. The cost function consists in minimizing the number of haplotypes used,

$$\text{minimize} \sum x_v \tag{1}$$

This model is referred to as *TIP* [7]. A more efficient model is *RTIP*, which introduces one key simplification. If genotype g_i can be explained by pair of haplotypes (h_a, h_b), such that both h_a and h_b cannot explain any other genotype, then the pair of haplotypes (h_a, h_b) needs not to be considered

for explaining g_i. If all pairs are discarded for a genotype g_i, then it suffices to pick any pair for explaining g_i.

5.2. Polynomial-Size ILP Models

One alternative to the exponential models is the *PolyIP* model, which is polynomial in the number of sites m and population size n [8, 1], with a number of constraints and variables, respectively, in $\Theta(n^2 m)$ and $\Theta(n^2 + n\,m)$. The PolyIP model represents the $2\,n$ candidate haplotypes as sequences of Boolean variables, and then establishes conditions for the haplotypes to explain the corresponding genotypes, such that the total number of distinct haplotypes is minimized. Haplotypes are represented with Boolean variables $y_{i\,j}$, $1 \leq i \leq 2\,n$ and $1 \leq j \leq m$, i.e. m variables for each of the $2\,n$ candidate haplotypes.

First, the PolyIP model defines conditions on the sites, with $1 \leq i \leq n$ and $1 \leq j \leq m$:

$$\begin{array}{rl} y_{2i-1\,j} = 0 \text{ and } y_{2i\,j} = 0, & \text{if } g_{ij} = 0, \\ y_{2i-1\,j} = 1 \text{ and } y_{2i\,j} = 1, & \text{if } g_{ij} = 1, \\ y_{2i-1\,j} + y_{2i\,j} = 1 & \text{if } g_{ij} = 2 \end{array} \tag{2}$$

where $g_{ij} \in \{0, 1, 2\}$ denotes the possible values at each site. Second, the PolyIP model defines conditions for identifying different haplotypes, with $1 \leq i, l \leq 2n$ and $1 \leq j \leq m$. Boolean variable $d_{i\,l}$ is defined such that $d_{i\,l} = 1$ if $h_i \neq h_l$. The resulting conditions become:

$$\begin{array}{l} y_{i\,j} - y_{l\,j} \leq d_{i\,l} \\ y_{l\,j} - y_{i\,j} \leq d_{i\,l} \end{array} \tag{3}$$

If at least one site of h_i and h_l differs, then $d_{i,l}$ needs to be assigned to value 1.

Third, the model introduces the x_i variables denoting whether h_i is different from all previous haplotypes h_l, where $1 \leq l < i$, and defines conditions on these variables. Boolean variable x_i is defined such that $x_i = 1$ if h_i is unique with respect to the previous haplotypes. Thus, if h_i is unique, then $\sum_{l=1}^{i-1} d_{l\,i} = i - 1$; otherwise $\sum_{l=1}^{i-1} d_{l\,i} < i - 1$. As a result, the condition on variable x_i becomes:

$$x_i \geq 2 - i + \sum_{l=1}^{i-1} d_{l\,i} \tag{4}$$

Finally, the cost function consists in minimizing the number of different haplotypes:

$$\text{minimize} \sum_{i=1}^{2n} x_i \tag{5}$$

A number of optimizations have been proposed to the basic PolyIP model [1], with the purpose of pruning the search space to be handled by the ILP solver. More recently, an alternative polynomial-size ILP model, *HybridIP*, was proposed [2], and represents a hybrid between the RTIP and the PolyIP models. Nonetheless, no significant improvements were achieved by HybridIP compared to PolyIP.

6. Solving HIPP with SAT

An alternative to solving HIPP with ILP is to use SAT. Current SAT solvers are characterized by being extremely fast at solving real world problem instances, mainly due to the capacity of learning new constraints whenever the search reaches a dead-end, as well as to very efficient data structures. SAT-based approaches for the HIPP problem were proposed recently in the SHIPs tool [11, 12], and allowed remarkable performance improvements over the existing ILP-based models.

The SAT-based HIPP solution algorithm starts from a lower bound lb on the number of haplotypes necessary to explain the set of genotypes; a trivial value for lb is 1. The algorithm searches for the smallest value r such that there exists a set $\mathcal{H}$ of haplotypes with $r = |\mathcal{H}|$, which explain all genotypes in $\mathcal{G}$. Observe that the value of r is guaranteed to satisfy $lb \leq r \leq 2\,n$, since a solution with $2\,n$ haplotypes is guaranteed to exist. For each value of r considered, a CNF formula φ^r is created, and a SAT solver is invoked.

In what follows the same indexes will be used throughout: i ranges over the genotypes and j over the sites, with $1 \leq i \leq n$ and $1 \leq j \leq m$, where n is the number of genotypes and m is the number of sites. In addition, r candidate haplotypes are considered, each with m sites, and with $1 \leq r \leq 2n$. An additional index k is associated with haplotypes, such that $1 \leq k \leq r$. As a result, $h_{kj} \in \{0, 1\}$ denotes the j^{th} site of haplotype k.

For a given value of r, the SHIPs model considers r haplotypes and seeks to associate two haplotypes (possibly corresponding to the same haplotype) with each genotype g_i, where $1 \leq i \leq n$. The Boolean variables used by SHIPs are depicted in Figure 3. For each genotype g_i the model uses *selector* variables for selecting which haplotypes are used for explaining g_i. Since the genotype is to be explained by *two* haplotypes, the model uses two sets, a and b, of r selector variables, respectively s^a_{ki} and s^b_{ki} with $k = 1, \ldots, r$. Hence, genotype g_i is explained by haplotypes h_{k_1} and h_{k_2} if $s^a_{k_1 i} = 1$ and $s^b_{k_2 i} = 1$. Clearly, g_i is also explained by the same haplotypes if $s^a_{k_2 i} = 1$ and $s^b_{k_1 i} = 1$.

We can now derive the conditions for the SHIPs model:

- If a site g_{ij} is 0 (resp. 1), and if haplotype k is selected for explaining genotype i, either by the a or the b representative, then the value of haplotype k at site j *must* be 0 (resp. 1). In CNF, if site g_{ij} is 0, then the model includes $(\neg s^a_{ki} \vee \neg h_{kj}) \wedge (\neg s^b_{ki} \vee \neg h_{kj})$, and if site g_{ij} is 1, then the model includes $(\neg s^a_{ki} \vee h_{kj}) \wedge (\neg s^b_{ki} \vee h_{kj})$, in both cases for $k = 1, \ldots, r$.

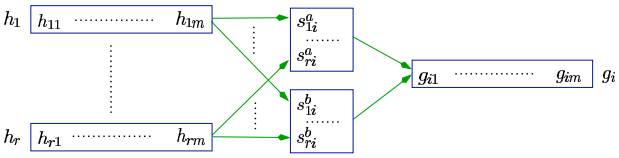

Figure 3. Boolean variables used in SHIPs

- Otherwise, one requires that the haplotypes explaining the genotype g_i have opposing values at site i. This is done by creating a variable $t_{ij} \in \{0,1\}$ such that site j of the haplotype selected by the a representative selector assumes the same value as t_{ij}, and site j of the haplotype selected by the b representative selector assumes the complementary value of t_{ij}. As a result the model requires $(h_{kj} \vee \neg t_{ij} \vee \neg s^a_{ki}) \wedge (\neg h_{kj} \vee t_{ij} \vee \neg s^a_{ki}) \wedge (h_{kj} \vee t_{ij} \vee \neg s^b_{ki}) \wedge (\neg h_{kj} \vee \neg t_{ij} \vee \neg s^b_{ki})$ for $k = 1, \ldots, r$. Observe that h_{kj} equals t_{ij} if $s^a_{ki} = 1$, and h_{kj} equals $\neg t_{ij}$ if $s^b_{ki} = 1$.

Clearly, for each genotype g_i, and for a or b, it is necessary that exactly one haplotype is used, and so exactly one selector variable can be assigned value 1. This can be captured with the following cardinality constraints:

$$\left(\sum_{k=1}^{r} s^a_{ki} = 1\right) \wedge \left(\sum_{k=1}^{r} s^b_{ki} = 1\right) \tag{6}$$

These cardinality constraints can be encoded in CNF in linear space, by introducing additional auxiliary variables [11, 12]. In current implementations, auxiliary variables are not handled differently by SAT solvers. However, the special handling of these variables may be considered in the future, as it has been recently shown that not branching on these variables, as well as not learning constraints with these variables, leads in general to more robust performances [15].

Besides the basic model outlined above, SAT-based haplotyping requires the inclusion of a number of effective techniques, including lower bounds and identification of symmetries [11] (see Section 4.3). More recent work addressed using local search algorithms for improving lower bounds in SAT-based approaches for the HIPP problem [13]. Another sucessful approch used answer set programming (ASP) [4]. Similarly to SHIPs, this ASP-based approach uses a SAT solver as a search engine. SAT solvers have also been recently used for solving the HIPP problem for non-diploid organisms [17].

7. Solving HIPP with PBO

The success of solving HIPP with SAT motivated considering other Boolean-based decision and optimization procedures. One very successful approach is based on using Pseudo-Boolean Optimization (PBO) in a tool called RPoly [5, 6].

The organization of RPoly is similar to the organization of PolyIP: two haplotypes are associated with each genotype, and conditions which capture when a different haplotype is used for explaining a given genotype are defined. With no surprise, the generated PBO formulas are much larger than the generated SAT formulas for a given HIPP problem instance. Whereas the PBO approach assumes the worst case for which the number of required haplotypes is twice the number of genotypes, the SAT approach incrementally increases the number of required haplotypes starting from a lower bound.

Despite the similarities, RPoly has a few key differences with respect to PolyIP. First, the set of variables is different. Instead of associating a variable with each site of each haplotype, RPoly only associates variables with heterozygous sites (since the value of haplotypes in the other sites is known beforehand, and so can be implicitly assumed). In addition, each used variable describes the possible pairs of values for the corresponding heterozygous site.

In practice, the model associates two haplotypes, h^a_i and h^b_i, with each genotype g_i, and these haplotypes are required to explain g_i. Moreover, the model associates a variable t_{ij} with each heterozygous site (i,j) (i.e. with $g_{ij} = 2$). Hence, $t_{ij} = 1$ indicates that $h^a_{ij} = 1$ and $h^b_{ij} = 0$, whereas $t_{ij} = 0$ indicates that $h^a_{ij} = 0$ and $h^b_{ij} = 1$. The value of h^a_i and h^b_i at homozygous sites j is implicitly assumed.

This alternative definition of the variables associated with the sites of genotypes reduces the number of variables by a factor of 2. In addition, the model only creates variables for heterozygous sites, and so the number of variables associated with sites equals the total number of heterozygous sites. As a result, the conditions provided by (2) are eliminated. It is interesting to observe that this definition of the variables associated with sites follows the SHIPs model [11, 12].

Finally, another key modification is that the candidate haplotypes for each genotype are related with candidate haplotypes for other genotypes only if the two genotypes are *compatible*. Clearly, incompatible genotypes are guaranteed not to be explained by the same haplotype.

The proposed modification implies the use of two additional sets of variables. Variable $x^{pq}_{i_1 i_2}$, with $p, q \in \{a, b\}$ and $1 \leq i_2 < i_1 \leq n$, is 1 if the p haplotype of genotype i_1 and the q haplotype of genotype i_2 are different. Clearly, if genotypes i_1 and i_2 are incompatible, then the value of $x^{pq}_{i_1 i_2}$ is 1 for the four possible combinations of p and q. Moreover, two genotypes i_1 and i_2 are related only with respect to sites j such that either g_{i_1} or g_{i_2} is heterozygous at that site. In addition, the model uses variables to denote

when one of the haplotypes associated with a given genotype is different from all previous haplotypes. Hence, u_i^p, with $p \in \{a, b\}$ and $1 \leq i \leq n$, is 1 if haplotype p of genotype i is different from all previous haplotypes.

The conditions on the u_i^p variables are based on the conditions for the x_i variables for the PolyIP model:

$$\bigwedge_{1 \leq k < i} (x_{i\,k}^{p\,a} \wedge x_{i\,k}^{p\,b}) \rightarrow u_i^p \tag{7}$$

The conditions on the $x_{i_1\,i_2}^{p\,q}$ variables are all of the following form, for all $1 \leq j \leq m$:

$$\neg(R \leftrightarrow S) \rightarrow x_{i_1\,i_2}^{p\,q} \tag{8}$$

Where the predicates R and S depend on the values of the sites (i_1, j) and (i_2, j), and on which of the haplotypes is considered, i.e. either a or b. Observe that $1 \leq i_2 < i_1 \leq n$, $1 \leq j \leq m$, and $p, q \in \{a, b\}$. Accordingly, the R and S predicates are defined as follows:

- If $g_{i_1\,j} \neq 2$, then $R = (g_{i_1\,j} \leftrightarrow (q \leftrightarrow a))$ and $S = t_{i_2\,j}$.
- If $g_{i_2\,j} \neq 2$, then $R = (g_{i_2\,j} \leftrightarrow (p \leftrightarrow a))$ and $S = t_{i_1\,j}$.
- If $g_{i_1\,j} = 2 \wedge g_{i_2\,j} = 2$, then $R = \neg(p \leftrightarrow q)$ and $S = \neg(t_{i_1\,j} \leftrightarrow t_{i_2\,j})$.

Finally, the cost function is given by:

$$\text{minimize} \sum_{i=1}^{n} (u_i^a + u_i^b) \tag{9}$$

The proposed simplifications to PolyIP model [1] yield significant performance improvements, even when the two models are solved with a PB solver [5]. More recently, a number of improvements to RPoly were proposed [6]. Similarly to SHIPs, one of the proposed improvements is the integration of lower bounds (see Section 4.3).

One useful feature of the RPoly tool is to be able to deal with unspecified genotype sites. Genotyping procedures often leave a percentage of missing genotype positions, and so haplotype inference tools need to be able to deal with missing sites. RPoly can handle SNPs with unspecified values, inferring the values for the missing sites and still guaranteeing a parsimonious solution. Two Boolean variables are associated with each missing site to represent the four possible values for the haplotypes: two homozygous values (one for each allele) and two heterozygous values (one for each haplotype phase). The constraints for unspecified genotype sites are similar to the constraints for heterozygous genotype sites.

Class	# Instances	*min*SNPs	*max*SNPs	*min*GENs	*max*GENs
SU1	100	104	173	80	90
SU2	100	156	188	89	90
SU3	100	128	182	87	90
SU-100kb	29	14	20	34	88
Total	329	14	188	34	90

Table 1. Classes of instances: number of SNPs and genotypes

8. Practical Experience

This section illustrates the behaviour of the models described above in a challenging set of 329 problem instances [1]. These instances were generated to evaluate phasing algorithms [14]. Table 1 characterizes the instances giving the number of instances for each class, as well as the minimum and maximum number of SNPs (*minSNPs* and *maxSNPs*) and genotypes (*minGENs* and *maxGENs*) for the instances of each class.

A comparison of alternative approaches for solving the HIPP problem is summarized in Figure 4. The HIPP solvers RTIP [7], PolyIP [1], HybridIP [2], SHIPs [12] and RPoly [6] were considered [2]. All HIPP solvers were run on a Intel Xeon 5160 server (3.0GHz, 1333Mhz, 4GB) running Red Hat Enterprise Linux WS 4.

The run times for each solver were sorted and plotted, the cutoff point being 1000 seconds. As the figure shows, the ILP approaches are significantly less efficient than the SAT/PBO approaches. The ILP approaches are able to solve less than 30% of the problem instances. SHIPs is able to solve around 81% of the problem instances, whereas RPoly is able to solve more than 94% of the problem instances.

This section also illustrates the effectiveness of the bounding techniques described in Section 4. For this study, only the instances that have been solved by at least one of the solvers have been taken into account. (This procedure has eliminated 12 of the 329 instances). Otherwise it would not be possible to compare the values of the computed bounds with the optimal solution.

Figure 5 provides a comparison between the lower bound and the HIPP solution. For around 30% of the instances, the lower bound computes the exact HIPP solution. Moreover, for the majority of the instances (more precisely 65%) the difference between the lower bound and the HIPP solution is less than or equal to 5.

The evaluation of the upper bound computation is summarized in Figure 6. For 16% of the instances, the upper bound algorithm computes the exact HIPP solution. In addition, for 65% of the instances the difference between

[1] Available from *http://www.stats.ox.ac.uk/∼marchini/phaseoff.html*.

[2] The results were obtained with the tools provided by the authors, except for the RTIP tool. This tool was provided by the authors of PolyIP and HybridIP. To our best knowledge, the author of RTIP has not made the software available.

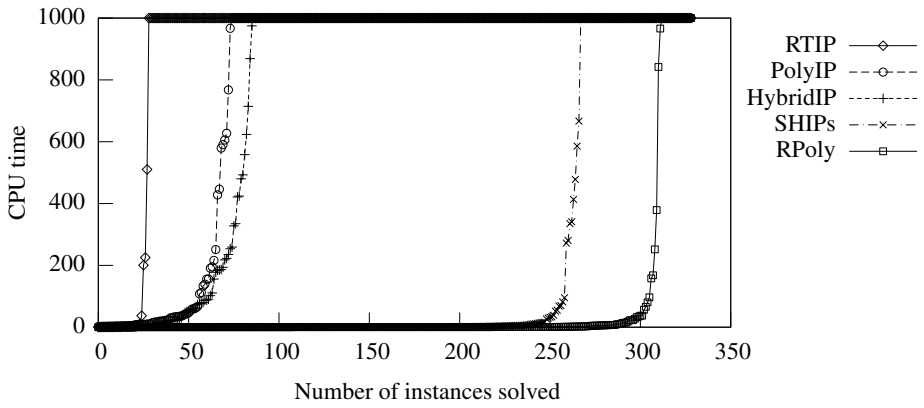

Figure 4. Relative Performance of HIPP solvers

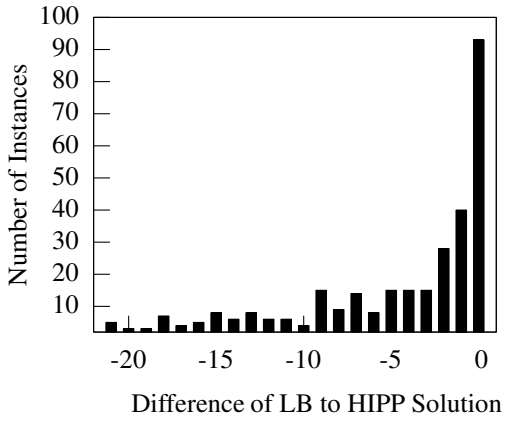

Figure 5. Quality of the Lower Bound

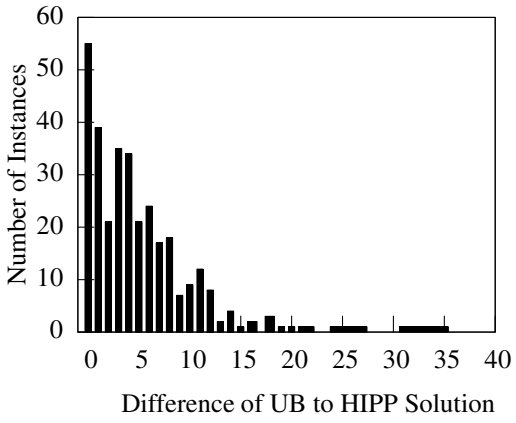

Figure 6. Quality of the Upper Bound

the computed upper bound and the HIPP solution is less or equal to 5, and for 88% the difference is less than or equal to 10.

Finally, Figure 7 compares the lower and upper bound values obtained for each instance [3]. For this plot the whole set of 329 instances has been evaluated. We may observe that for more than 10% of the instances both values are exactly the same. This means that computing lower and upper bounds suffices to solve these problem instances, i.e. no search is required. In addition, the difference between the upper bound and the lower bound is more than 10 for less than half of the instances, thus predictably not requiring much time to be solved.

[3]These results are not as impressive as the ones reported in the original publications [12, 16] because here only a subset of very hard instances is being considered.

9. Research Directions

Pure parsimony has been shown in the past to be an accurate approach for haplotype inference [18]. Accuracy is measured by the correct association between genotypes and explaining haplotypes. Although it is not possible in general to know the precise solution for the haplotype inference problem, there are very well-studied sets of genotypes for which the solution is known. This solution is often obtained using different generations from the same population.

In the future, additional criteria should be taken into account to further improve the accuracy of the HIPP algorithms. This is motivated by the fact that in general, and for a single instance, the number of solutions satisfying the pure parsimony criterion can be significantly large.

The reason for a large number of solutions is that although the HIPP criterion imposes a constraint on the number of haplotypes in the solution, the same set of haplotypes

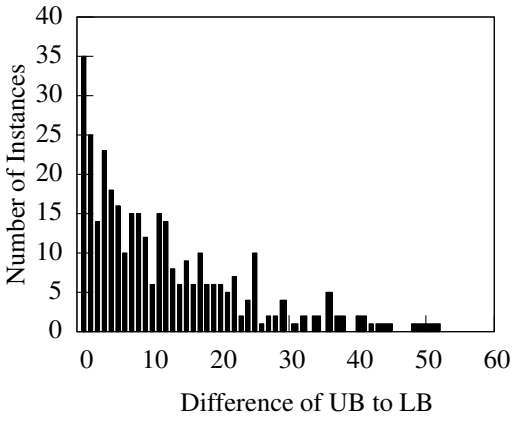

Figure 7. Quality of the Bounds

can be used in different ways to explain the genotypes. In addition, there can be solutions with different sets of haplotypes that still have minimum size.

To illustrate this issue, we have performed an extensive evaluation for a specific instance: SU100kb.25, which has 34 genotypes and 15 sites. This is one of the smallest instances and therefore it is easier to get all HIPP solutions. The SU100kb.25 instance has 48 parsimonious solutions with 17 haplotypes each. 14 out of 17 haplotypes are common to all HIPP solutions. The remaining 3 haplotypes are picked from a set of 7 haplotypes and are used in general to explain only one genotype. If we compare each pair of HIPP solutions, we observe that out of the 1128 solution pairs, 72 pairs have exactly the same haplotypes, 384 pairs differ in 1 haplotype, 480 pairs differ in 2 haplotypes and 192 pairs differ in 3 haplotypes.

The measures given above make it clear that there are significantly different solutions, and therefore picking the first solution to be found may not be the best option in terms of accuracy. Future research directions should consider using a criterion to choose the most accurate solution between all possible HIPP solutions.

Acknowledgments

This work is partially funded by Microsoft under contract 2007-017 of the Microsoft Research PhD Scholarship Programme, and by FCT under research projects POSC/EIA/ 61852/2004 and PTDC/EIA/64164/2006 and PhD grant SFRH/BD/28599/2006.

References

[1] D. Brown and I. Harrower. A new integer programming formulation for the pure parsimony problem in haplotype analysis. In *Workshop on Algorithms in Bioinformatics (WABI'04)*, pages 254–265, 2004.

[2] D. Brown and I. Harrower. Integer programming approaches to haplotype inference by pure parsimony. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(2):141–154, 2006.

[3] A. G. Clark. Inference of haplotypes from pcr-amplified samples of diploid populations. *Molecular Biology and Evolution*, 7(2):111–122, 1990.

[4] E. Erdem and F. Türe. Efficient haplotype inference with answer set programming. In *AAAI Conference on Artificial Intelligence*, pages 436–441, July 2008.

[5] A. Graça, J. Marques-Silva, I. Lynce, and A. Oliveira. Efficient haplotype inference with pseudo-Boolean optimization. In *Algebraic Biology*, pages 125–139, 2007.

[6] A. Graça, J. Marques-Silva, I. Lynce, and A. Oliveira. Efficient haplotype inference with combined CP and OR techniques. In *CPAIOR'08*, pages 308–312, 2008.

[7] D. Gusfield. Haplotype inference by pure parsimony. In *14th Annual Symposium on Combinatorial Pattern Matching (CPM'03)*, pages 144–155, 2003.

[8] B. Halldórsson, V. Bafna, N. Edwards, R. Lippert, S. Yooseph, and S. Istrail. A survey of computational methods for determining haplotypes. In *DIMACS/RECOMB Satellite Workshop on Computational Methods for SNPs and Haplotype Inference*, pages 26–47, 2004.

[9] R. Hudson. Gene genealogies and the coalescent process. *Oxford Survey of Evolutionary Biology*, 7:1–44, 1990.

[10] G. Lancia, C. M. Pinotti, and R. Rizzi. Haplotyping populations by pure parsimony: complexity of exact and approximation algorithms. *INFORMS Journal on Computing*, 16(4):348–359, 2004.

[11] I. Lynce and J. Marques-Silva. Efficient haplotype inference with Boolean satisfiability. In *AAAI Conference on Artificial Intelligence*, pages 104–109, July 2006.

[12] I. Lynce and J. Marques-Silva. Haplotype inference with Boolean satisfiability. *International Journal on Artificial Intelligence Tools*, 17(2):355–387, April 2008.

[13] I. Lynce, J. Marques-Silva, and S. Prestwich. Boosting haplotype inference with local search. *Constraints*, 13(1):155–179, 2008.

[14] J. Marchini, D. Cutler, N. Patterson, M. Stephens, E. Eskin, E. Halperin, S. Lin, Z. Qin, H. Munro, G. Abecassis, P. Donnelly, and International HapMap Consortium. A comparison of phasing algorithms for trios and unrelated individuals. *American Journal of Human Genetics*, 78:437–450, 2006.

[15] J. Marques-Silva and I. Lynce. Towards robust CNF encodings of cardinality constraints. In *Principles and Practice of Constraint Programming (CP)*, September 2007.

[16] J. Marques-Silva, I. Lynce, A. Graça, and A. Oliveira. Efficient and tight upper bounds for haplotype inference by pure parsimony using delayed haplotype selection. In *13th Portuguese Conference on Artificial Intelligence (EPIA 07)*, pages 621–632, 2007.

[17] J. Neigenfind, G. Gyetvai, R. Basekow, S. Diehl, U. Achenbach, C. Gebhardt, J. Selbig, and B. Kersten. Haplotype inference from unphased snp data in heterozygous polyploids based on SAT. *BMC Genomics*, 9(356), 2008.

[18] L. Wang and Y. Xu. Haplotype inference by maximum parsimony. *Bioinformatics*, 19(14):1773–1780, 2003.