Efficient Haplotype Inference with Pseudo-Boolean Optimization

Ana Graça[1], João Marques-Silva[2], Inês Lynce[1], and Arlindo L. Oliveira[1]

[1] IST/INESC-ID, Technical University of Lisbon, Portugal
{assg,ines}@sat.inesc-id.pt,aml@inesc-id.pt
[2] School of Electronics and Computer Science, University of Southampton, UK
jpms@ecs.soton.ac.uk

Abstract. Haplotype inference from genotype data is a key computational problem in bioinformatics, since retrieving directly haplotype information from DNA samples is not feasible using existing technology. One of the methods for solving this problem uses the pure parsimony criterion, an approach known as Haplotype Inference by Pure Parsimony (HIPP). Initial work in this area was based on a number of different Integer Linear Programming (ILP) models and branch and bound algorithms. Recent work has shown that the utilization of a Boolean Satisfiability (SAT) formulation and state of the art SAT solvers represents the most efficient approach for solving the HIPP problem.
Motivated by the promising results obtained using SAT techniques, this paper investigates the utilization of modern Pseudo-Boolean Optimization (PBO) algorithms for solving the HIPP problem. The paper starts by applying PBO to existing ILP models. The results are promising, and motivate the development of a new PBO model (RPoly) for the HIPP problem, which has a compact representation and eliminates key symmetries. Experimental results indicate that RPoly outperforms the SAT-based approach on most problem instances, being, in general, significantly more efficient.

Key words: haplotype inference, pure parsimony, pseudo-Boolean optimization

1 Introduction

The causes of many common human diseases remain, to this day, largely unknown. Since genetic inheritance is one of the major risk factors for the large majority of diseases, the study of genetic variation in human populations represents one of the critical steps towards a better understanding of the mechanisms of disease.

Although a number of heritable disorders that depend on the variation of one single location in one single gene are known, common diseases usually depend on the combined effects of many different factors, in a number of different genes.

The study of the effects of particular variations of genes is simplified by the fact that, in many cases, there exists a strong correlation between the allele

present in a particular single nucleotide polymorphism (SNP) and other nearby sites. A given combination of alleles in one chromosome is termed a haplotype, and the deviation from independence that exists between alleles is known as linkage disequilibrium (LD).

For genetic inheritable diseases that are due to a combination of allele values in nearby loci, identifying common haplotypes in the population represents a key first step towards the understanding of the pathogenesis of disease. However, current genotyping methods do not provide haplotype information, which is essential for detailed analysis of the mechanisms of disease.

At a given position for which an individual is heterozygous (i.e., inherited different alleles at a given locus), it is technologically not feasible, in general, to identify the particular chromosome that contains each allele. Additional information can be obtained by genotyping the parents, but significant uncertainty remains. Efficient methods for haplotype inference that can handle large volumes of data are therefore crucial, in order to make adequate use of the results of ongoing efforts like the HapMap project [17], an effort that aims at making available genotype and haplotype information of a significant sample of the human population.

Although a number of different methods has been proposed for the problem of haplotype inference, the Pure-Parsimony criterion [6, 10, 7] represents a well known approach. Haplotype Inference by Pure-Parsimony (HIPP) aims at finding a solution to the problem that minimizes the total number of distinct haplotypes required. The problem of finding such a solution is APX-hard (and, therefore, NP-hard) [10]. Experimental results [6, 18] have shown that the accuracy of the HIPP approach is comparable with the one obtained with other approaches. However, until recently, HIPP inference methods were severely limited on the size of the problems they could handle. Recently, a SAT based approach for this problem, SHIPs [11, 12], has shown that the use of effective constraint satisfaction methods leads to an efficient solution of this problem.

Motivated by these results, this paper explores an alternative approach. Existing ILP models only have Boolean variables and, therefore, can be solved with Pseudo-Boolean Optimization (PBO) solvers [5, 13]. Hence, this paper starts by considering the utilization of PBO solvers instead of standard ILP solvers. The results are very promising, being competitive with SHIPs. These results motivate the development of a new PBO model (RPoly) for the HIPP problem, which is based on the PolyIP model [1, 8] and, in addition, breaks key symmetries and yields a significantly more compact representation. The results show that RPoly is, in general, more efficient than SHIPs, and capable of solving more problem instances in a given time limit.

This paper is organized as follows. First we introduce the haplotype inference by pure parsimony problem. Afterwards, we describe the two main contributions of the paper: (1) how to solve HIPP ILP models using PBO and (2) how to optimize the existing polynomial model. Finally, we conclude and suggest future research work.

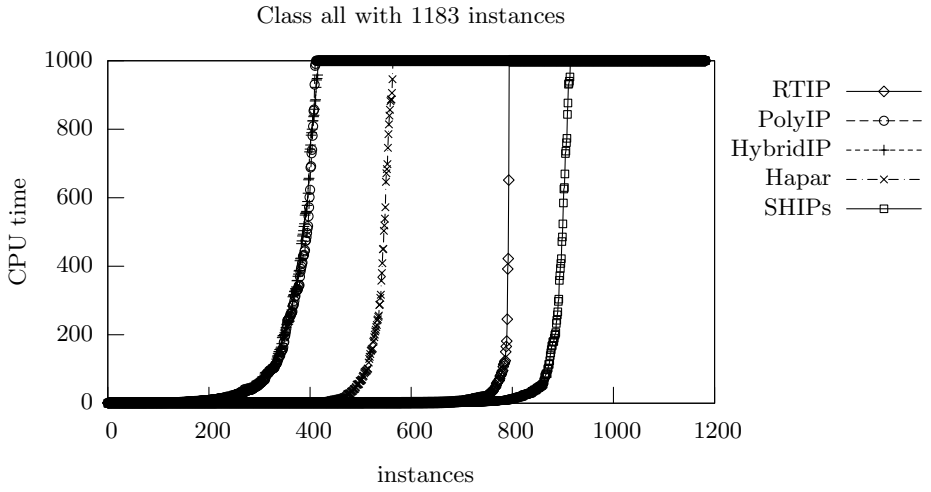

Fig. 1. Relative performance of HIPP solvers

2 Haplotype Inference by Pure Parsimony

A *haplotype* is the genetic constitution of an individual chromosome. The underlying data that forms a haplotype is generally viewed as the set of SNPs in a given region of a chromosome. Normal cells of diploid organisms contain two haplotypes, one inherited from each parent. The *genotype* represents the conflated data of the two haplotypes. The value of a particular SNP may be A, B or A/B, depending on whether the organism is homozygous with allele A, homozygous with allele B or heterozygous.

Starting from a set of genotypes, the haplotype inference by pure parsimony problem consists in finding a minimum set of haplotypes that can be used to derive, by pairwise combinations, the given set of genotypes.

Given a set $\mathcal{G}$ of n genotypes, each of length m, the haplotype inference problem consists in finding a set $\mathcal{H}$ of $2 \cdot n$ haplotypes, not necessarily different, such that for each genotype $g_i \in \mathcal{G}$ there is at least one pair of haplotypes (h_j, h_k), with h_j and $h_k \in \mathcal{H}$ such that the pair (h_j, h_k) explains g_i. The variable n denotes the number of individuals in the sample, and m denotes the number of SNP sites. g_i denotes a specific genotype, with $1 \leq i \leq n$. Furthermore, g_{ij} denotes a specific site j in genotype g_i, with $1 \leq j \leq m$.

Without loss of generality, we may assume that the values of the two possible alleles of each SNP are always 0 or 1. Value 0 represents the wild type and value 1 represents the mutant. A haplotype is then a string over the alphabet {0,1}. Moreover, genotypes may be represented by extending the alphabet used for representing haplotypes to {0,1,2}. Homozygous sites are represented by the

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

Class	# Instances	*min*SNPs	*max*SNPs	*min*GENs	*max*GENs
ms	380	4	57	9	94
phasing	329	14	188	34	90
hapmap	24	4	29	5	68
biological	450	4	77	4	49
Total	1183	4	188	4	94

values 0 or 1, depending on whether both haplotypes have value 0 or 1 at that site, respectively. Heterozygous sites are represented by value 2.

The HIPP problem is to find a minimum-size set $\mathcal{H}$ of haplotypes that explain all genotypes in $\mathcal{G}$. For example, consider the set of genotypes: 2120, 2120 and 1221. There are solutions for this example that use six distinct haplotypes, but solution 0100/1110, 0100/1101, 1011/1101 uses only four distinct haplotypes.

Two strings (denoting genotypes or haplotypes) are *incompatible* if and only if the strings have at least one site where one string has value 1 and the other string has value 0. Otherwise the strings are said to be *compatible*.

A comparison of the performance of alternative approaches to the HIPP problem is summarized in Figure 1. A universe of 1183 problem instances is used, from which 854 instances were taken from [12] and the remaining (harder) instances are described by Schaffner [15] and correspond to the SU-100kb, SU1, SU2 and SU3 classes available from `http://www.stats.ox.ac.uk/~marchini-/phaseoff.html`. All problem instances were simplified in a preprocessing step, according to what has been suggested in [2]: duplicated genotypes and sites were removed, as well as complemented sites. For each class, Table 1 gives the number of instances, and the minimum and maximum number of SNPs and genotypes, respectively, after removing duplicated genotypes and duplicated and complemented sites. The *ms* class includes the uniform and nonuniform classes of instances that have been used in [2] but extended with additional, more complex, problem instances. The *phasing* instances correspond to the instances described in [15] which were generated to evaluate phasing algorithms. The *hapmap* class of instances is also the one used in [2]. Finally, the instances for the *biological* class were generated from publicly available data (e.g. [14, 4, 3, 9]).

The HIPP solvers RTIP [6], PolyIP [1], HybridIP [2], Hapar [18] and SHIPs [12] were considered[1]. The run times for each solver were sorted and plotted, the cut-off point being 1000 seconds. All results shown were obtained on a 1.9 GHz AMD Athlon XP with 1GB of RAM running RedHat Linux. For the ILP-based HIPP solvers, the ILP package used was CPLEX version 7.5. As can be concluded, SHIPs is the HIPP tool capable of solving the largest number of problem instances. SHIPs aborts 268 problem instances out of 1183 instances, whereas RTIP aborts 389 instances, Hapar aborts 619 instances, HybridIP aborts 767 instances and PolyIP aborts 771 instances. Nonetheless, we should note that

[1] All results were obtained with the tools provided by the authors.

95% of the problem instances aborted by RTIP were aborted due to memory exhaustion. Hence, RTIP may be competitive for solving some problem instances but it is not a robust solver.

3 Solving ILP HIPP Models with PBO

This section reviews existing ILP models for the HIPP problem [7, 10]. In addition, the section includes results using a modern Pseudo-Boolean Optimization (PBO) solver instead of a standard ILP solver.

In a pseudo-Boolean formula, variables have Boolean domains and constraints are linear inequalities with integer coefficients,

$$\sum c_i x_i \geq n \quad c_i, n \in \mathbb{Z}, x_i \in \{0, 1\}. \tag{1}$$

For example, $x + 2\,y - z \geq 2$ is a pseudo-Boolean constraint (also denoted as PB-constraint). From an ILP point of view, PB-constraints can be seen as a specialization of ILP where all variables are Boolean. This problem formulation is also known as *0-1 integer programming*. From a SAT point of view, PB-constraints can be seen as a generalization of clauses. Furthermore, a pseudo-Boolean formula can be extended with an optimization function.

3.1 Exponential-Size ILP Models

The first ILP model proposed for the HIPP problem, *RTIP* [6], has linear space complexity on the number of possible haplotypes and, therefore, it is exponential on the number of given genotypes.

A Boolean variable $y_{i,u}$ is associated with each pair u of haplotypes that can explain a given genotype g_i, and denotes whether this pair of haplotypes is used for explaining g_i. A cardinality constraint,

$$\sum_u y_{i,u} = 1, \tag{2}$$

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,

$$y_{i,u} \rightarrow x_v, \tag{3}$$

for each haplotype in the pair. The cost function is used to minimize the number of haplotypes used,

$$\text{minimize} \sum x_v. \tag{4}$$

This model corresponds to the *TIP* model [6]. The *RTIP* (Reduced TIP) model introduces one essential simplification. If the pair of haplotypes for a variable

$y_{i,u}$ is such that they are not part of any other pair of haplotypes, then the $y_{i,u}$ variable and the related x_v variables can be removed from the formulation. A key drawback of the RTIP model is that the number of candidate haplotypes grows exponentially with the number of heterozygous sites. Hence, RTIP does not scale for large problem instances.

The RTIP model inspired a branch-and-bound algorithm to the HIPP problem, known as *Hapar* [18].

3.2 Polynomial-Size ILP Models

A more recent ILP model, *PolyIP* [1], is polynomial in the number of sites m and population size n, with a number of constraints and variables, respectively, in $\Theta(n^2m)$ and $\Theta(n^2 + n\,m)$. The PolyIP model represents the $2 \cdot n$ 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 $2 \cdot n$ candidate haplotypes.

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

$$\begin{aligned} y_{2i-1\,j} = 0 \text{ and } y_{2i\,j} = 0, \text{ if } g_{ij} = 0, \\ y_{2i-1\,j} = 1 \text{ and } y_{2i\,j} = 1, \text{ if } g_{ij} = 1, \\ y_{2i-1\,j} + y_{2i\,j} = 1 \text{ if } g_{ij} = 2, \end{aligned} \tag{5}$$

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 $1 \leq l \leq i \leq 2n$ and $1 \leq j \leq m$. Boolean variable $d_{l\,i}$ is defined such that $d_{l\,i} = 1$ if $h_i \neq h_l$. The resulting conditions become

$$\begin{aligned} y_{i\,j} - y_{l\,j} \leq d_{l\,i}, \\ y_{l\,j} - y_{i\,j} \leq d_{l\,i}. \end{aligned} \tag{6}$$

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

$$x_i \geq 2 - i + \sum_{l=1}^{i-1} d_{l\,i}. \tag{7}$$

Finally, the cost function minimizes the number of different haplotypes,

$$\text{minimize } \sum_{i=1}^{2n} x_i. \tag{8}$$

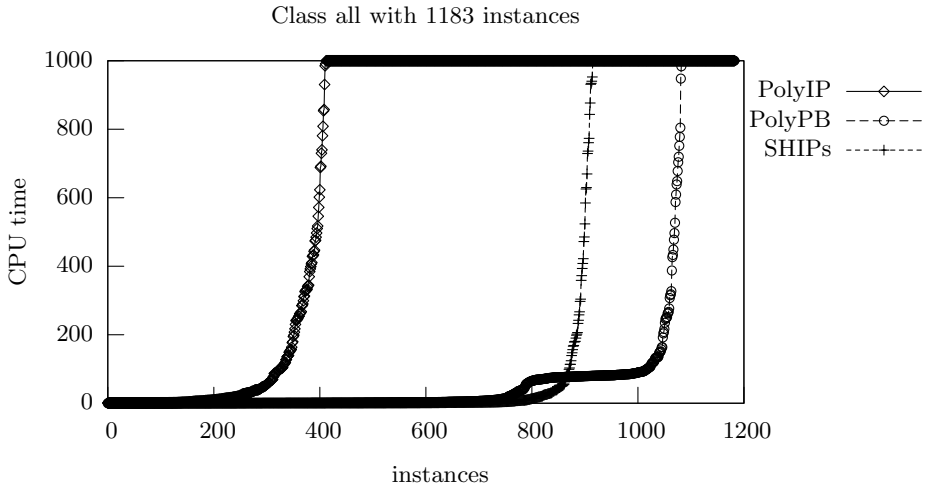

Fig. 2. Relative performance of PolyIP, PolyPB and SHIPs

A number of optimizations have been proposed to the basic PolyIP model [1], with the purpose of improving the quality of the LP relaxation step of standard ILP solvers, and therefore pruning the search space to be handled by the ILP solver.

More recently, the same authors introduced a new polynomial-size formulation, *HybridIP* [2], representing a hybrid of the RTIP and PolyIP formulations. Nevertheless, existing experimental results (see Figure 1) suggest that the performance of the two polynomial models does not differ significantly.

3.3 ILP vs. PBO Solvers

As is clear from the description of the ILP models, all variables are Boolean and all coefficients are integer. Hence, the HIPP ILP models are also PBO models, and so PBO solvers can be considered. The results summarized in Figure 1 indicate that the performances of the PolyIP and HybridIP models are similar. Moreover, the RTIP model is known to be inadequate for larger problem instances, due to the exponential growth of the model in the number of heterozygous sites per genotype. As a result, this section only evaluates the performance of the PolyIP model using a PBO solver (hereafter referred to as PolyPB). The PBO solver MiniSAT+ [5] is used on all reported PBO results. Although other PBO solvers analyzed in [13] were considered, MiniSAT+ was by far the most efficient.

MiniSAT+ handles PB-constraints through translation to SAT without modifying the SAT procedure itself. In addition, the objective function is satisfied

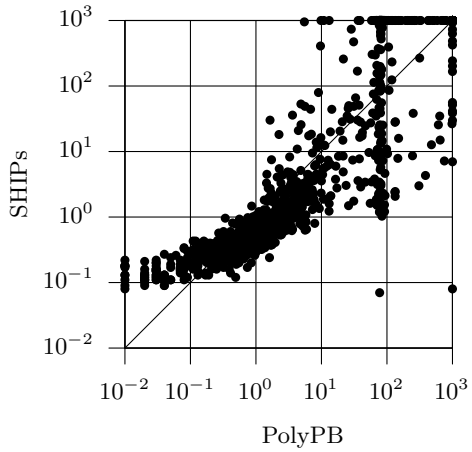

Fig. 3. Run times for PolyPB and SHIPs

by iteratively calling the SAT solver where for each new iteration the objective function is updated until the problem is unsatisfiable. For example, given a minimization problem with an objective function $f(x)$, MiniSAT+ first runs the solver on the set of constraints (without considering the objective function) to get an initial solution $f(x_0) = k$. Then it adds the constraint $f(x) < k$ and runs the solver again. If the problem is unsatisfiable, then k is the optimum solution. If not, the process is repeated with the new smaller solution. Observe that translating to SAT results in an approach that is particularly suited for problems that are almost pure SAT. Indeed, this is the case for the HIPP problem. Hence, one may expect to get a faster procedure with MiniSAT+ than by applying a native PBO solver, not optimized towards propositional SAT.

Figure 2 compares the PolyIP model using the CPLEX solver, the PolyPB model using the PBO solver MiniSAT+ and the SHIPs solver on the 1183 problem instances described in Table 1 for a timeout of 1000 seconds. Clearly, PolyPB outperforms SHIPs in terms of the number of instances solved. Although both solvers are able to solve the majority of the 1183 problem instances within 1000 seconds, PolyPB only aborts 100 instances whereas SHIPs aborts 268 instances. Observe that PolyIP is significantly worse, aborting 771 out of 1183 instances.

In addition, Figure 3 provides a scatter plot with the run time for PolyPB and SHIPs on each of the problem instances with a timeout of 1000 seconds. For most problem instances SHIPs is faster than PolyPB; PolyPB is faster than SHIPs on 454 out of 1183 instances, with many of these instances being solved in less than one second. Nonetheless, this group of instances for which PolyPB is faster than SHIPs also includes 184 instances that PolyPB is able to solve and SHIPs aborts. On the other hand that there are only 16 instances that SHIPs is able to solve and PolyPB aborts. As a result, we can conclude that PolyPB is

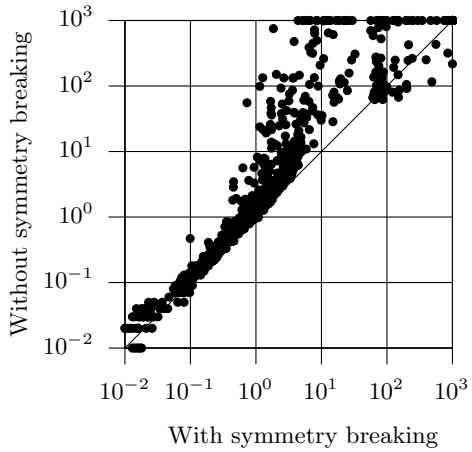

Fig. 4. Run times for PolyPB with and without symmetry breaking

more robust than SHIPs. Finally, there are still 84 instances that both solvers are unable to solve within 1000 seconds.

4 RPoly: an Optimized PolyPB Model

Although the results shown in the previous section are promising, it is possible to further optimize the PolyPB model. Indeed, SHIPs is still showing a better performance in a large number of problem instances, which motivates the incorporation of some of the SHIPs model features into the PolyPB model. This section addresses optimizations to the PolyPB model with the main goal of reducing the run times.

These optimizations are two-fold: (1) the elimination of key symmetries and (2) the reduction of the size of the model. It is well-know that the SHIPs model would not be competitive if it was not for some specific optimizations, which include breaking key symmetries. Symmetries are broken by adding constraints to the model. We have also observed that the PBO instances generated with the PolyPB model are significantly larger than the SAT instances generated with the SHIPs model. The number of constraints in the PBO model can be up to an order of magnitude larger than the number of constraints in the SAT model, whereas the number of variables in the PBO model can be up to a factor of 3 larger than the number of variables in the SAT model.

The resulting model is referred to as *Reduced Poly model (RPoly)*.

4.1 Eliminating Key Symmetries

A key technique for pruning the search space is motivated by observing the existence of symmetry in the problem formulation. Clearly, given a solution

to a HIPP problem were a genotype g_i is explained by the pair of haplotypes (h_{2i-1},h_{2i}), the same genotype g_i may also be explained by the pair of haplotypes (h_{2i},h_{2i-1}). Eliminating this symmetry significantly reduces the number of solutions and consequently reduces the search space.

In practice, this kind of symmetry is eliminated by adding additional constraints to the model, which guarantee that the elements in a pair of haplotypes are lexicographically ordered. Hence, for each site g_{ij} in a genotype g_i we must force the following:

- If $g_{i\,j} = 2$ and $g_{i\,j'} \neq 2$ $(\forall j' : j' < j)$, then $y_{2i-1\,j} - y_{2i\,j} < 0$.

Figure 4 compares the performance of the PolyPB model with and without symmetry breaking constraints. Clearly, with a few exceptions (72 out of 1183 instances), eliminating symmetries accelerates the performance of the PBO solver. The new model is faster than the PolyPB model for 90% of the instances and up to 2 orders of magnitude. This result comes as no surprise, given the success of the same technique when implemented in the SHIPs model. This result is indeed significant, as the new model only aborts 47 instances, whereas the PolyPB model aborts 100 instances.

4.2 Reducing the Model

The organization of RPoly follows the organization of PolyIP: two haplotypes are associated with each genotype, and conditions are defined which capture when a different haplotype is used for explaining a given genotype. However, RPoly has a few key differences. 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_i^a and h_i^b, with each genotype g_i, and these haplotypes are required to explain g_i. Moreover, the model associates a variable $t_{i\,j}$ with each heterozygous site (i, j) (i.e. with $g_{i\,j} = 2$). Hence, $t_{i\,j} = 1$ indicates that $h^a_{i\,j} = 1$ and $h^b_{i\,j} = 0$, whereas $t_{i\,j} = 0$ indicates that $h^a_{i\,j} = 0$ and $h^b_{i\,j} = 1$ [2]. The value of h_i^a and h_i^b 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, therefore, the number of variables associated with sites equals the total number of heterozygous sites. As a result, the conditions provided by expression (5) are eliminated. It should also be mentioned that this definition of the variables associated with sites follows the SHIPs model [11, 12].

[2] Hence, the symmetry in a pair of haplotypes is broken by considering that $t_{ij} = 0$ for the first heterozygous site g_{ij} of each genotype g_i.

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 $x_{i_1\,i_2}^{p\,q}$, 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 incompatible. Clearly, if genotypes i_1 and i_2 are incompatible, then the value of $x_{i_1\,i_2}^{p\,q}$ 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,

$$\bigwedge_{1 \leq k < i} (x_{i\,k}^{p\,a} \wedge x_{i\,k}^{p\,b}) \rightarrow u_i^p. \tag{9}$$

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

$$\neg(R \leftrightarrow S) \rightarrow x_{i_1\,i_2}^{p\,q}, \tag{10}$$

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

$$\text{minimize} \sum_{i=1}^{n} (u_i^a + u_i^b). \tag{11}$$

The proposed modifications result in significantly smaller PBO problem instances. Figure 5 compares the number of terms for the PolyPB and the RPoly models. The results are consistent and show that the number of terms in RPoly is a factor of 5 to 10 smaller than in PolyPB. Albeit not shown, the number of variables in RPoly can be up to a factor of 3 smaller than the number of variables in PolyPB. We should note that the *phasing* class of instances exhibits a different behavior: most of these instances have around 10^7 terms in the PBO model with symmetry breaking. The number of terms in RPoly is not reduced for a constant factor, as it is for the other classes of instances. These instances

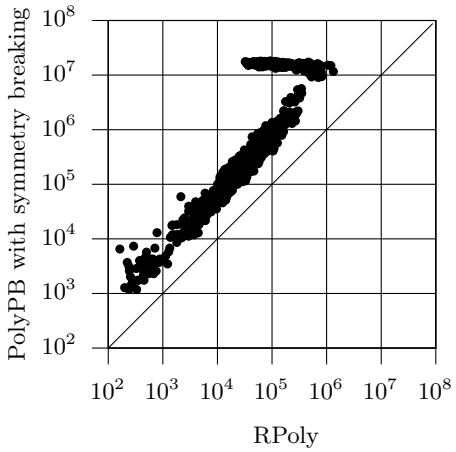

Fig. 5. Number of terms for PolyPB and RPoly

have a higher number of incompatible genotypes when compared with the other classes of instances. Hence, the impact of the reduced model is much more significant. For the same reason, the impact on the run times is also more significant (see Figure 6 where the run time for the *phasing* instances using the PBO model with symmetry breaking is around 10^2 seconds). As a result, for these instances RPoly can outperform PolyPB by two orders of magnitude.

Finally we evaluate the effect of the reductions described above with respect to the run times. Figure 6 compares the PolyPB model extended with symmetry breaking constraints and the RPoly model, both using the PBO solver MiniSAT+, on the set of 1183 problem instances and with a timeout of 1000 seconds. With a few exceptions (28 out of 1183 instances), RPoly is consistently faster than PolyPB, and the speedup can reach 2 orders of magnitude. The few exceptions where RPoly is slower are explained by the branching heuristics used by MiniSAT+, which, in some cases, may not select the most adequate variables to branch on.

4.3 RPoly vs. SHIPs

In this section we measure the progress made with this work, by comparing the SHIPs model [12] with the RPoly model. The RPoly model is based on the PolyIP model but uses a PBO solver, MiniSAT+, and introduces key optimizations: the elimination of symmetries between the elements within a pair of haplotypes and the reduction on the size of the model.

Although both RPoly and SHIPs use SAT-based technology, the two approaches differ. Whereas SHIPs considers an increasing number of haplotypes until a solution is found, RPoly considers $2 \cdot n$ haplotypes, where n is the num-

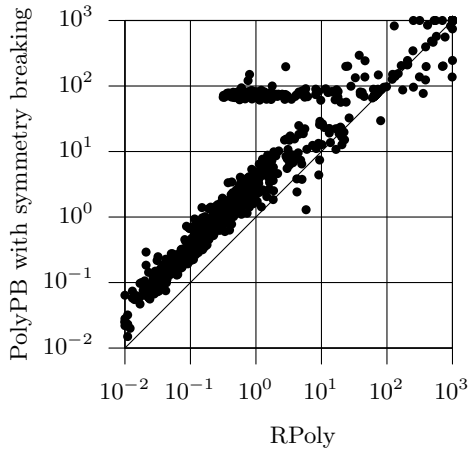

Fig. 6. Run times for PolyPB with symmetry breaking and RPoly

ber of genotypes, and iteratively reduces the number of different haplotypes until a solution with a minimum number of different haplotypes is found.

Figure 7 compares the RPoly model using the PBO solver MiniSAT+ and the SHIPs solver. For a small number of problem instances (52 out of 1183) SHIPs is faster than RPoly, and the speedup can reach 2 orders of magnitude. However, for most problem instances (1089 out of 1183), RPoly is faster than SHIPs. It should be observed that SHIPs is, in general, slower on very easy problem instances, essentially due to the initial setup time [11]. Nevertheless, the results also clearly show that RPoly is significantly more robust than SHIPs. RPoly aborts on a significantly smaller number of instances, being able to solve more than 96% of the problem instances. Finally, observe that only two instances aborted by RPoly can be solved by SHIPs.

5 Conclusions and Future Work

This paper studies the application of modern PBO solvers to the HIPP problem. By replacing the CPLEX ILP solver with the PBO solver MiniSAT+ [5], the existing PolyIP model [1] is shown to be competitive with the state-of-the-art method, SHIPs [12], being in general more robust. These results motivated the development of a new ILP model for the HIPP problem, RPoly, which entails a number of improvements to the basic PolyIP model inspired by SHIPs. The results for RPoly are significantly more promising than for PolyIP: RPoly is most often faster than SHIPs and is also significantly more robust, aborting only on a small number of problem instances (observe that, with two exceptions, SHIPs also aborts all of these instances).

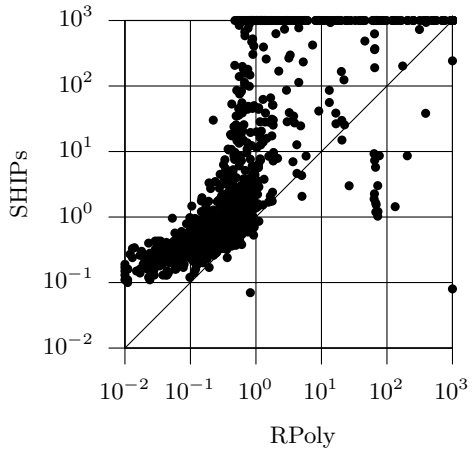

Fig. 7. Run times for RPoly and SHIPs

The results indirectly suggest that the performance improvements obtained with SHIPs [11, 12] are to a large extent explained by the efficiency of modern SAT solvers. Indeed, SAT-inspired PBO solvers obtain extremely good results with PolyIP and with RPoly, which are PBO models that differ significantly from the SHIPs SAT-based approach. In addition, the different PBO models provide a new, relevant, and essentially endless, source of challenging real problem instances for PBO solvers.

Despite the promising results obtained using MiniSAT+ with the RPoly model, several challenges remain. A number of problem instances cannot be solved by any HIPP solver. In addition, larger HIPP instances are expected to be significantly more challenging.

Finally, we should mention that having a competitive HIPP solver allows us to extend the pure parsimony approach with some ideas which are on the basis of other haplotype inference approaches. This will enable us to develop parsimony-based methods that explicitly incorporate genetic models (e.g. as in Phase [16]), with the objective of improving the accuracy of the reconstructed haplotypes.

Acknowledgments This work is partially supported by Fundação para a Ciência e Tecnologia under research project POSC/EIA/61852/2004 and PhD grant SFRH/BD/28599/2006, and by INESC-ID under research project SHIPs.

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)*, volume 3240, pages 254–265, October 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. M. J. Daly, J. D. Rioux, S. F. Schaffner, T. J. Hudson, and E. S. Lander. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29:229–232, October 2001.
4. C. M. Drysdale, D. W. McGraw, C. B. Stack, J. C. Stephens, R. S. Judson, K. Nandabalan, K. Arnold, G. Ruano, and S. B. Liggett. Complex promoter and coding region β_2-adrenergic receptor haplotypes alter receptor expression and predict in vivo responsiveness. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 97, pages 10483–10488, September 2000.
5. N. Eén and N. Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26, March 2006.
6. D. Gusfield. Haplotype inference by pure parsimony. In *14th Annual Symposium on Combinatorial Pattern Matching (CPM'03)*, volume 2676, pages 144–155, February 2003.
7. D. Gusfield and S. Orzach. *Handbook on Computational Molecular Biology*, volume 9 of *Chapman and Hall/CRC Computer and Information Science Series*, chapter Haplotype Inference. CRC Press, December 2005.
8. B. Halldórsson, V. Bafna, N. Edwards, R. Lippert, S. Yooseph, and S. Istrail. A survey of computational methods for determining haplotypes. In *Proceedings of the First RECOMB Satellite on Computational Methods for SNPs and Haplotype Inference*, volume 2983 of *LNBI*, pages 26–47, February 2004.
9. D. L. Kroetz, C. Pauli-Magnus, L. M. Hodges, C. C. Huang, M. Kawamoto, S. J. Johns, D. Stryke, T. E. Ferrin, J. DeYoung, T. Taylor, E. J. Carlson, I. Herskowitz, K. M. Giacomini, and A. G. Clark. Sequence diversity and haplotype structure in the human ABCD1 (MDR1, multidrug resistance transporter). *Pharmacogenetics*, 13:481–494, November 2003.
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, March 2004.
11. I. Lynce and J. Marques-Silva. Efficient haplotype inference with Boolean satisfiability. In *National Conference on Artificial Intelligence (AAAI)*, July 2006.
12. I. Lynce and J. Marques-Silva. SAT in bioinformatics: Making the case with haplotype inference. In *International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 136–141, August 2006.
13. V. Manquinho and O. Roussel. The first evaluation of Pseudo-boolean solvers (PB'05). *Journal on Satisfiability, Boolean Modeling and Computation*, 2:103–143, March 2006.
14. M. J. Rieder, S. T. Taylor, A. G. Clark, and D. A. Nickerson. Sequence variation in the human angiotensin converting enzyme. *Nature Genetics*, 22:59–62, May 1999.
15. S. Schaffner, C. Foo, S. Gabriel, D. Reich, M. Daly, and D. Altshuler. Calibrating a coalescent simulation of human genome sequence variation. *Genome Reasearch*, 15:1576–1583, November 2005.
16. M. Stephens, N. Smith, and P. Donelly. A new statistical method for haplotype reconstruction. *American Journal of Human Genetics*, 68:978–989, February 2001.
17. The International HapMap Consortium. A haplotype map of the human genome. *Nature*, 437:1299–1320, October 2005.
18. L. Wang and Y. Xu. Haplotype inference by maximum parsimony. *Bioinformatics*, 19(14):1773–1780, March 2003.