Generic ILP *vs* Specialized 0-1 ILP for Haplotype Inference

Ana Graça[1], Inês Lynce[1], João Marques-Silva[2], 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 is an important and computationally challenging problem in genetics. A well-known approach to haplotype inference is pure parsimony (HIPP). Despite being based on a simple optimization criterion, HIPP is a computationally hard problem. Recent work has shown that approaches based on Boolean satisfiability namely pseudo-Boolean optimization (PBO), are very effective at tackling the HIPP problem. Extensive work on PBO-based HIPP approaches has been recently developed. Considering that the PBO problem, also known as 0-1 ILP problem, is a particular case of the integer linear programming (ILP) problem, generic ILP solvers can be considered. This paper compares the performance of PBO and ILP solvers on a variety of HIPP models. We conclude that specialized PBO solvers are more suitable than generic ILP solvers.

1 Introduction

Understanding genetic differences between human beings is a crucial step towards the diagnosis and prevention of genetic diseases. Haplotype inference is a key problem to solve for achieving this goal, since haplotypes include most of the information about human genetic variations. A well-known haplotype inference approach is the pure parsimony (HIPP) which, among the possible solutions, chooses the one with the smallest number of distinct haplotypes [7].

Former work on the HIPP problem was mainly based on integer linear programming (ILP) [7,2,3]. Afterwards, Boolean satisfiability (SAT) [9] and pseudo-Boolean optimization (PBO) [5] have been used to solve the problem. Recently, PBO HIPP-based approaches have been improved, generating further reduced models [6]. Considering that PBO is a particular case of ILP, existing PBO models can also be solved by generic ILP solvers.

This work compares the performance of different HIPP models described in the literature [2,5,6], using different PBO solvers [11,10,4,1] and the generic ILP solver CPLEX. To the best of our knowledge, such a comparison has never been made in the past. This paper aims at performing a comprehensible evaluation of different models and solvers. The analysis of the experimental results gives insights to select the most appropriate modelling techniques depending on the kind of solver being used.

The paper is organized as follows. The next section describes the HIPP problem. Section 3 details recent PBO HIPP models, namely RPoly [5] and the recent improvements to the model [6]. Afterwards, on section 4, experimental results comparing ILP

and PBO approaches to the HIPP problem are presented. Finally, the paper concludes in section 5.

2 Haplotype Inference by Pure Parsimony

Single nucleotide polymorphisms (SNPs) correspond to sites in the DNA sequence where mutations have occurred and which represent a significant variability on the population. Haplotypes can be seen as a sequence of SNPs highly correlated, within a single chromosome. It is technically difficult to obtain haplotypes directly. Instead, genotypes, which correspond to the mixed data of two haplotypes on homologous chromosomes, are experimentally obtained. The haplotype inference problem consists in finding the set of haplotypes which originated a given set of genotypes.

Considering that mutations are rare, we may assume that each SNP can only have two values. Each haplotype is therefore represented by a binary string with size $m \in \mathbb{N}$, where 0 represents the wild type and 1 represents the mutant type. Each site of the haplotype h_i is represented by h_{ij} ($1 \leq j \leq m$). Each genotype is represented by a string, with size m, over the alphabet $\{0, 1, 2\}$, and each site of the genotype g_i is represented by g_{ij}. Each genotype is explained by two haplotypes. A genotype $g_i \in \mathcal{G}$ is explained by a pair of haplotypes (h_i^a,h_i^b) such that

$$g_{ij} = \begin{cases} h_{ij}^a \text{ if } h_{ij}^a = h_{ij}^b \\ 2 \quad \text{if } h_{ij}^a \neq h_{ij}^b \end{cases}, \tag{1}$$

with $1 \leq j \leq m$. A genotype site g_{ij} with either value 0 or 1 is a homozygous site, whereas a site with value 2 is a heterozygous site.

Definition 1. *Given a set $\mathcal{G}$ of n genotypes each with size m, the haplotype inference problem consists in finding a set of haplotypes $\mathcal{H}$, such that each genotype $g_i \in \mathcal{G}$ is explained by two haplotypes $h_i^a, h_i^b \in \mathcal{H}$.*

For each genotype g with k heterozygous sites, there are 2^{k-1} pairs of haplotypes that can explain g. For example, genotype $g_i = 202$ can be explained either by haplotypes (000,101) or by haplotypes (001,100). Several approaches to the haplotype inference problem have been suggested. Given that individuals from the same population share many haplotypes, the pure parsimony approach searches for a solution with the smallest number of distinct haplotypes.

Definition 2. *The haplotype inference by pure parsimony (HIPP) problem consists in finding a solution to the haplotype inference problem which minimizes the number of distinct haplotypes [7].*

Example 1. Consider the set of genotypes $\mathcal{G}$: $g_1 = 022$, $g_2 = 221$ and $g_3 = 222$. There are solutions using 6 different haplotypes $\mathcal{H}_1$: $h_1^a = 001, h_1^b = 010, h_2^a = 011$, $h_2^b = 101, h_3^a = 000$ and $h_3^b = 111$. However the HIPP solution only requires 4 distinct haplotypes $\mathcal{H}_2$: $h_1^a = 011, h_1^b = 000, h_2^a = 011, h_2^b = 101, h_3^a = 011$ and $h_3^b = 100$.

It has been shown that the HIPP problem is NP-hard [8].

3 ILP/PBO-based HIPP Models

With a few notable exceptions [12], early work on the HIPP problem used models based on integer linear programming [7,2,3]. The original ILP model, *RTIP* [7], has exponential space complexity on the number of heterozygous sites because, in the worst case, it requires the enumeration of all possible pairs of haplotypes that can explain each genotype. Afterwards, two polynomial ILP models, *PolyIP* [2] and *HybridIP* [3], were proposed [1].

Recently, a very competitive SAT-based approach, *SHIPs* [9], suggested an incremental algorithm that, starting from a clique-based lower bound on the number of required haplotypes, models the problem into SAT and searches for a HIPP solution. If no solution is found, the lower bound is incremented by one and a new SAT instance is generated. When a solution is found, the minimum number of haplotypes is given by the value of the lower bound. More recent approaches use pseudo-Boolean optimization models [5,6]. These models represent an improvement in terms of the efficiency of HIPP solvers.

The *Reduced Poly* model (RPoly) [5] proposed a number of simplifications to the Poly model. The RPoly model associates two haplotypes (h_i^a, h_i^b) with each genotype g_i, for $1 \leq i \leq n$. A variable t_{ij} is associated with each heterozygous site g_{ij}, such that $t_{ij} = 1$ if $h_{ij}^a = 1$ and $h_{ij}^b = 0$, whereas $t_{ij} = 0$ if $h_{ij}^a = 0$ and $h_{ij}^b = 1$.

Another key issue in the RPoly's formulation is the notion of incompatibility. Two genotypes are incompatible if they cannot be explained by a common haplotype, or equivalently, genotypes g_i and g_k, are incompatible if there exists j ($1 \leq j \leq m$) such that $g_{ij} + g_{kj} = 1$. Otherwise, they are said to be compatible. For candidate haplotypes h_i^p and h_k^q, with $p, q \in \{a, b\}$ and $1 \leq k < i \leq n$, a variable $x_{i\,k}^{p\,q}$ is defined, such that, $x_{i\,k}^{p\,q} = 1$ if haplotype h_i^p of genotype g_i and haplotype h_k^q of genotype g_k are different. If two genotypes are incompatible, then they cannot share an explaining haplotype, and consequently, for the four possible combinations of p and q, $x_{i\,k}^{p\,q} = 1$.

Finally, in order to count the number of distinct haplotypes used, variables u_i^p are defined such that $u_i^p = 1$ if haplotype h_i^p, which explains genotype g_i, is different from all the haplotypes which explain genotypes g_k, with $k < i$. The conditions on variables u_i^p are

$$\sum_{1 \leq k < i \,;\, q \in \{a,b\}} x_{i\,k}^{p\,q} - u_i^p \leq 2i - 3, \tag{2}$$

with $p \in \{a, b\}$ and $1 \leq i \leq n$. The objective function minimizes the sum of variables u_i^p.

An improved RPoly model was recently proposed [6]. This new model, *NRPoly*, integrates the SHIPs clique-based lower bound in the RPoly model and extends the model with additional constraints. The components of the SHIPs lower bound allow both fixing the value of some of the u_i^p variables and also avoiding generating the constraints involving fixed u_i^p variables [9]. Moreover, the order in which the genotypes are considered must reflect the order in which the genotypes are used in the lower bound [6]. In

[1] Throughout the paper we will remove the suffix *IP* and use only *Poly* and *Hybrid*.

practice, the integration of SHIP's lower bound allows fixing the value of many u_i^p variables and, as several constraints need not be generated, allows significantly reducing the size of the model.

The second optimization is related with a key simplification of the RTIP model, which consists in not considering pairs of haplotypes when both of them do not explain more than one genotype. Actually, if a genotype g_i is not incompatible with all other genotypes, then at least one of the haplotypes that explain g_i must explain other genotype. For each genotype $g_i \in \mathcal{G}$ compatible with at least one more genotype in $\mathcal{G}$, the following constraint is generated,

$$\sum_{k>i\,;\,p,q\in\{a,b\}\,;\,\kappa(k,i)} x_{k\,i}^{p\,q} + u_i^a + u_i^b \leq 4\mathcal{K} + 1, \tag{3}$$

where predicate $\kappa(k,i)$ is defined true if g_k and g_i are compatible and $\mathcal{K}$ is the cardinality of the set $\{g_k \in \mathcal{G} : k > i \wedge \kappa(k,i)\}$.

The last optimization consists in adding cardinality constraints on the values of variables x. For many combinatorial problems, adding cardinality constraints to the model can prune the search space, helping the solver to find a solution. Clearly, two different genotypes g_i and g_k cannot be explained by the same pair of haplotypes, and then g_i and g_k can be at most explained by one haplotype in common. Therefore, for each pair of distinct genotypes g_i and g_k ($k < i$), if g_i and g_k are compatible and non-homozygous, then

$$\sum_{p,q\in\{a,b\}} x_{i\,k}^{p\,q} \geq 3. \tag{4}$$

4 Generic ILP *vs* Specialized 0-1 ILP for the HIPP problem

In this section we compare the relative performance of discrete optimization HIPP models using different 0-1 ILP solvers and a generic ILP solver. A considerable number of HIPP models and solvers are evaluated. To the best of our knowledge, such comparation has never been performed so far.

4.1 Experimental Setup

An extensive evaluation, using 1183 problem instances [5] including real and synthetic data, has been performed. The solvers used were MiniSat+ [4], Pueblo [11] version 1.5, the latest version of BSOLO [10], PBS4 [1], glpPB release 0.2 and CPLEX version 11.0 (www.ilog.com/products/cplex/). The results were obtained on an Intel Xeon 5160 server (3.0GHz, 4MB RAM) running Red Hat Enterprise Linux WS 4. The timeout for each instance was set to 1000 seconds.

4.2 Results

The Poly, RPoly and NRPoly models were adapted to be run by the five different 0-1 ILP solvers (Minisat+, Pueblo, BSOLO, PBS4 and glpPB) and the generic ILP solver

Table 1. Number of instances aborted (out of 1183) for each model and solver (timeout 1000s)

	PBO solver					**ILP Solver**
Model	MiniSat+	Pueblo	BSOLO	PBS4	glpPB	CPLEX
Poly	82	251	486	605	1091	705
RPoly	36	127	290	326	723	234
NRPoly	18	55	120	108	611	249

Table 2. Number of instances aborted (out of 1183) using CPLEX, on each ILP model (timeout 1000s)

Model	RTIP	Poly	Hybrid	RPoly	NRPoly
# aborted	378	707	717	234	249

CPLEX. All solvers are then being evaluated on exactly the same models. Table 1 provides a summary of the results obtained, with the number of instances aborted (out of 1183) for each model.

Clearly, the best performing solver for each of the three models is MiniSat+. In general, MiniSat+, Pueblo, BSOLO and PBS4 solvers outperform CPLEX. The only exception is with the RPoly model, where CPLEX solves more instances than both BSOLO and PBS4.

For the results shown, the Poly model used is a re-implementation of the model described in [2]. The original Poly model gives similar results, aborting 707 instances instead of the 705 shown, using CPLEX. Even though the original Poly model was developed to be solved using CPLEX, the results suggest that most of the specialized 0-1 ILP solvers perform better for this model.

The glpPB solver is the worst performing solver for each of the three models (Poly, RPoly, NRPoly). glpPB is a ILP-based pseudo-Boolean solver, that uses the GNU linear programming kit (GLPK, www.gnu.org/software/glpk/). Hence, the glpPB ILP-based solver implements some of the techniques also used by CPLEX, but glpPB is not as optimized as CPLEX.

For all PBO solvers, NRPoly is shown to be more robust than the previous RPoly model. Solving the NRPoly model using MiniSat+, Pueblo, BSOLO or PBS4, reduces at least by 50% the number of instances not solved within 1000 seconds. Using the glpPB solver , the number of aborted instances is reduced in 15%. However, the generic ILP solver, CPLEX, does not benefit from the techniques introduced in the new model. Indeed, the NRPoly model aborts 15 instances more than the RPoly model, using CPLEX.

Table 2 summarizes the number of aborted instances for each model using CPLEX. For RTIP, Poly and Hybrid the same code used in [3], developed to be used with CPLEX, has been run. The RPoly and NRPoly models were adapted to be run by CPLEX. The number of instances aborted by the most recent models, RPoly and NRPoly, is much smaller than the number of instances aborted by the previous models, confirming that the new models are more robust. However, as already mentioned be-

Table 3. Number of instances aborted (out of 1183) by each version of NRPoly model using CPLEX

Model	RPoly	NRPoly *v1*	NRPoly *v2*	NRPoly
# aborted	234	258	257	249

fore, the most recent model, NRPoly, does not perform as well as the RPoly model, in contrast with PBO solvers.

In order to understand whether NRPoly performs worse than RPoly due to a particular feature of the NRPoly model, we analyzed the performance of NRPoly for each additional new technique included in this model. Table 3 presents the number of aborted instances for each NRPoly version. We call *NRPoly v1* to the version that only integrates the lower bound of SHIPs. *NRPoly v2* corresponds to the version with the lower bound of SHIPs and cardinality constraints on the x variables. The final version, that includes also the RTIP pruning, is simply *NRPoly*. As can be concluded, the integration of the lower bound of SHIPs is the reason why NRPoly performs worse than RPoly (24 more instances are aborted) when using CPLEX. In fact, both the integration of cardinality constraints and the RTIP pruning have been shown to help the CPLEX solver.

Finally, figure 1 provides a plot comparing RPoly using either MiniSat+ or CPLEX, and NRPoly using either MiniSat+ or CPLEX [2]. RPoly with MiniSat+ is more efficient than RPoly with CPLEX (36 vs. 234 aborted instances) and NRPoly with MiniSat+ is more efficient than NRPoly with CPLEX (18 vs. 249 aborted instances). The set of instances aborted using MiniSat+ is a subset of instances aborted by CPLEX. This result is not surprising given that Poly with MiniSat+ has been shown in the past to be more efficient than Poly with CPLEX [5]. However, taking into account that the two versions of Poly were not implemented by the same authors, this new comparison was deemed necessary.

5 Conclusions

This paper analyzes the performance of different generic and specialized ILP solvers on recently proposed HIPP models. Our experiments show that the SAT-based PBO solvers are, in general, more suitable than the state of the art generic ILP solver CPLEX. The experimental results confirm that the poor performance of CPLEX is a consequence of the ILP techniques used. Similar conclusions can be drawn for the ILP-based glpPB solver, which is the worst-performing ILP solver for the HIPP problem. glpPB uses some of the techniques used by CPLEX, but is significantly less optimized. Moreover, the results for CPLEX and glpPB suggest that similar results would be obtained in case a different ILP solver was considered.

Our conclusion is that, for the HIPP case and probably for other problems which can be naturally formulated as a 0-1 ILP problem, specific PBO solvers should be con-

[2] Each point in the plot corresponds to a problem instance, where the x-axis corresponds to the CPU time required by MiniSat+ and the y-axis corresponds to the CPU time required by CPLEX.

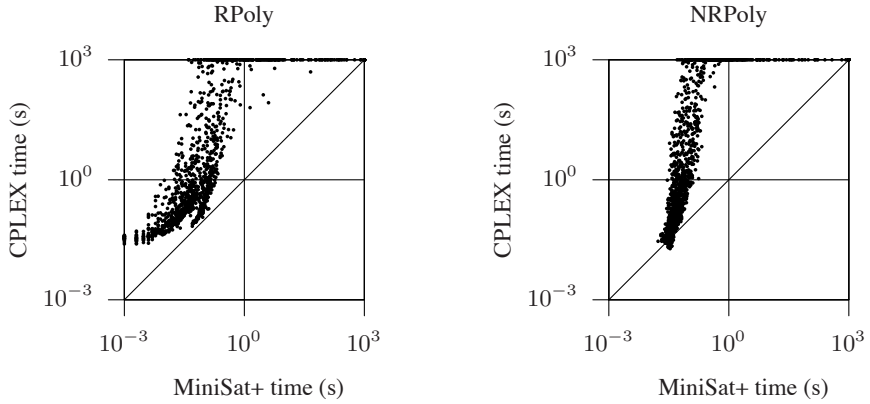

Fig. 1. CPU time results for RPoly and NRPoly

sidered. Furthermore, we observe that some modeling techniques used to optimize the PBO approaches do not produce improvements when the ILP solver CPLEX is used.

Acknowledgments

This work is partially supported by Fundação para a Ciência e Tecnologia under research projects SATPot (POSC/EIA/61852/2004) and SHIPs (PTDC/EIA/64164/2006) and PhD grant SFRH/BD/28599/2006, and by Microsoft under contract 2007-017 of the Microsoft Research PhD Scholarship Programme.

References

1. F. Aloul, A. Ramadi, I. Markov, and K. Sakallah. Generic ILP versus specialized 0-1 ILP: an update. In *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pages 450–457, 2002.
2. 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.
3. 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.
4. N. Eén and N. Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26, 2006.
5. A. Graça, J. Marques-Silva, I. Lynce, and A. Oliveira. Efficient haplotype inference with pseudo-Boolean optimization. In *Algebraic Biology 2007 (AB'07)*, pages 125–139, 2007.
6. A. Graça, J. Marques-Silva, I. Lynce, and A. Oliveira. Efficient haplotype inference with combined CP and OR techniques (short paper). In *5th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Problems (CPAIOR'08)*, 2008. Accepted for publication.

7. D. Gusfield. Haplotype inference by pure parsimony. In *14th Annual Symposium on Combinatorial Pattern Matching (CPM'03)*, pages 144–155, 2003.
8. 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.
9. I. Lynce and J. Marques-Silva. Efficient haplotype inference with Boolean satisfiability. In *National Conference on Artificial Intelligence (AAAI)*, 2006.
10. V. Manquinho and J. Marques-Silva. Effective lower bounding techniques for pseudo-Boolean optimization. In *Design, Automation and Test in Europe Conference and Exhibition (DATE'05)*, pages 660–665, 2005.
11. H. M. Sheini and K. A. Sakallah. Pueblo: A hybrid pseudo-Boolean SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:165–189, 2006.
12. L. Wang and Y. Xu. Haplotype inference by maximum parsimony. *Bioinformatics*, 19(14):1773–1780, 2003.