Efficient Haplotype Inference with Combined CP and OR Techniques

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 has relevant biological applications, and represents a challenging computational problem. Among others, pure parsimony provides a viable modeling approach for haplotype inference and provides a simple optimization criterion. Alternative approaches have been proposed for haplotype inference by pure parsimony (HIPP), including branch and bound, integer programming and, more recently, propositional satisfiability and pseudo-Boolean optimization (PBO). Among these, the currently best performing HIPP approach is based on PBO. This paper proposes a number of effective improvements to PBO-based HIPP, including the use of lower bounding and pruning techniques effective with other approaches. The new PBO-based HIPP approach reduces by 50% the number of instances that remain unsolvable by HIPP based approaches.

1 Introduction

Haplotype inference is a challenging computational problem, with a significant number of applications in genetics. Current DNA sequencing technology is not able to sequence independently the two copies of each chromosome which define the genetic inheritance of each diploid organism, such as humans. However, diagnosis and prevention of genetically related diseases requires, in many cases, the identification of the exact DNA sequences of each chromosome. This leads to the development of computational methods that can infer the haplotypes from the now easily obtained genotype information.

Over the last few years, Boolean satisfiability (SAT) and pseudo-Boolean optimization (PBO) techniques have been used to speed up one particular haplotype inference approach, based on pure parsimony [4]. Despite the success, the haplotype inference by pure parsimony (HIPP) problem is computationally hard, and there are several test cases that no HIPP solver is able to tackle. As a result, either alternative criteria or approximate algorithms are commonly used. With the objective of generalizing the use of HIPP solvers in haplotyping, it is important to increase the robustness of HIPP solvers, by increasing the number of instances HIPP solvers can solve efficiently. This paper pursues this objective, and combines CP and OR techniques that further reduce the search space, thus being able to solve some of the most difficult problem instances.

The paper is organized as follows. Section 2 introduces the HIPP problem. Section 3 describes the PBO-based HIPP approach, RPoly, and section 4 introduces the new techniques for improving the RPoly model. Afterwards, experimental results show that the new PBO model is able to solve a larger number of problem instances.

2 Haplotype Inference by Pure Parsimony (HIPP)

A haplotype is as a sequence of single nucleotide polymorphisms (SNPs) within a single chromosome. SNPs correspond to DNA nucleotides where mutations have occurred. Hence, for sites in the chromosome corresponding to SNPs we may either have the wild type (represented by 0) or the mutant type (represented by 1). Genotypes represent the conflated data contained in haplotypes. Each genotype is explained by two haplotypes. Unlike haplotypes, genotypes may be obtained using sequencing techniques.

Haplotype inference is the problem of identifying a set of haplotypes that may explain a given set of genotypes. A formal definition follows.

Definition 1. *Given a set of n genotypes $\mathcal{G}$, each genotype $g \in \mathcal{G}$ is represented by a string of size m over the alphabet $\{0, 1, 2\}$. The j^{th} element of the i^{th} genotype is referred to as g_{ij} with $1 \leq i \leq n$ and $1 \leq j \leq m$. Genotype g_i is heterozygous at site j if $g_{ij} = 2$ and is homozygous if $g_{ij} = 0$ or $g_{ij} = 1$. The* haplotype inference problem *consists in identifying a set of n pairs of haplotypes $\mathcal{H}$, not necessarily disjoint, with each haplotype h being represented by a string of size m over the alphabet $\{0, 1\}$, such that each pair of haplotypes explains a given genotype. A pair of haplotypes (h_i^a, h_i^b) is said to explain a genotype g_i ($g_i = h_i^a \otimes h_i^b$) if the following holds (with $1 \leq j \leq m$):*

$$\begin{array}{l} h_{ij}^a = h_{ij}^b = 0, \ \text{if } g_{ij} = 0; \\ h_{ij}^a = h_{ij}^b = 1, \ \text{if } g_{ij} = 1; \\ h_{ij}^a = 1 - h_{ij}^b, \ \text{if } g_{ij} = 2. \end{array}$$

It is clear that there is some freedom when selecting pairs of haplotypes for explaining genotypes with more than one site with value 2. For example, genotype 022 may be explained either by the pair of haplotypes (001,010) or by the pair of haplotypes (000,011). However, there is a biological motivation for selecting among the possible solutions to a set of genotypes the one with the smallest number of distinct haplotypes. Given that individuals from the same population have common ancestors and that mutations do not occur often, it is expected that individuals from the same population share a significant percentage of haplotypes.

Definition 2. *The approach that restricts the solutions to the haplotype inference problem such that the required number of haplotypes is minimum is called* pure parsimony *[4]. Finding a solution with a minimum number of haplotypes is a NP-hard problem [5].*

3 RPoly: A Pseudo-Boolean HIPP Model

The most well-known tools for solving the HIPP problem can be divided into four categories: (i) RTIP [4], PolyIP [1] and HybridIP [1] are integer linear programming (ILP) formulations, (ii) Hapar [8] is a branch and bound algorithm, (iii) SHIPs [6] is a SAT-based model for the HIPP problem and (iv) RPoly [3] is a pseudo-Boolean model.

The pseudo-Boolean optimization model, referred to as *Reduced Poly model (RPoly)* [3], is currently the best performing algorithm for the HIPP problem. RPoly is based on the PBO model for PolyIP and further enhanced with key optimizations.

The RPoly 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, RPoly associates a variable t_{ij} with each heterozygous site g_{ij}, such that $t_{ij} = 1$ indicates that $h_{ij}^a = 1$ and $h_{ij}^b = 0$, whereas $t_{ij} = 0$ indicates that $h_{ij}^a = 0$ and $h_{ij}^b = 1$. The values of h_i^a and h_i^b at homozygous sites are implicitly assumed.

Furthermore, let $x_{i\,k}^{p\,q}$, with $p, q \in \{a, b\}$ and $1 \leq k < i \leq n$, be 1 if haplotype p of genotype g_i and haplotype q of genotype g_k are different. The conditions on the $x_{i\,k}^{p\,q}$ variables are based on the values of variables t_{ij} and t_{kj} for heterozygous sites.

Moreover, two genotypes are said to be *incompatible* if there exists a site for which the value of one genotype is 0 and the other is 1; otherwise they are *compatible*. Clearly, candidate haplotypes for each genotype are related with candidate haplotypes for other genotypes only if the two genotypes are compatible. Then, incompatible genotypes g_i and g_k are guaranteed not to be explained by the same haplotype and so the value of $x_{i\,k}^{p\,q}$ is 1 for the four possible combinations of p and q.

In addition, the model uses variables u to denote whether 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 g_i is different from all previous haplotypes. Then, the conditions on the u_i^p variables are based on the conditions for the $x_{i\,k}^{p\,q}$ variables, with $1 \leq k < i$ and $q \in \{a, b\}$.

Finally, the cost function minimizes the number of distinct haplotypes used, which is given by the sum of variables u_i^p. The next section describes new improvements to the RPoly model, which allow significant additional performance improvements.

4 Optimizations to the RPoly Model

This section describes optimizations to the RPoly model, the state of the art HIPP solver. The resulting model is called New RPoly (NRPoly for short).

The first optimization consists in integrating the lower bounds of SHIPs [6, 7] in the NRPoly model. SHIPs is a SAT-based HIPP approach that, starting from a lower bound on the number of haplotypes, generates a SAT instance for each candidate number of haplotypes. SHIPs most recent lower bound procedure [7] provides a list of genotypes with an indication of the contribution of each genotype to the lower bound. Each genotype either contributes with +2, indicating that 2 new haplotypes will be required for explaining this genotype, or with +1, indicating that 1 new haplotype will be required for explaining this genotype.

In practice, for each genotype with an associated haplotype, the corresponding u variable, denoting whether a haplotype used for explaining a genotype is different from the haplotypes considered so far, is assigned value 1, and the clauses used for constraining the value of u *need not* be generated. The NRPoly model needs to be generated in such a way that the first genotypes correspond to genotypes used in the lower bound.

Similarly to the advantages of using lower bounds in SHIPs, the integration of lower bounds in NRPoly offers a few relevant advantages. First, several variables u become fixed with value 1, allowing the solver to focus on the remaining variables. Second, the size of the generated PBO problem instances is significantly reduced. The integration of lower bound information can reduce the generated PBO instances up to a factor of 3.

The second optimization is based on a key simplification introduced in the RTIP model [4], which consists in not considering all pairs of haplotypes that can explain a genotype. If a genotype can be explained by a pair of haplotypes such that none of these two haplotypes can explain any other genotype, then this pair of haplotypes needs not be considered.

Inspired by the pruning in RTIP, new constraints can be added to the NRPoly model. First, observe that each genotype that is not incompatible with all other genotypes must be explained by at least one haplotype that also explains some other genotype. Therefore, if a genotype g_i is explained by a pair of haplotypes (h_i^a, h_i^b) such that neither h_i^a nor h_i^b have been used to explain a genotype with lower index, then at least one of the haplotypes, h_i^a or h_i^b, must be used to explain one of the genotypes with higher index.

Consider genotypes compatible with at least one other genotype in $\mathcal{G}$. Define the predicate $\kappa(i,k)$ to be true if g_i and g_k are compatible. Formally, for all $1 \leq i \leq n$ such that g_i is compatible with at least another genotype in $\mathcal{G}$:

$$\text{If } u_i^a \wedge u_i^b, \text{ then } \exists_{k>i,\kappa(i,k)} \exists_{p,q \in \{a,b\}} \neg x_{k\,i}^{p\,q}. \tag{1}$$

Finally, an additional improvement consists in enriching the model with cardinality constraints on the x variables. For many combinatorial problems, adding new constraints to a model prunes the search and it is therefore likely to contribute to the solver being more efficient at finding solutions.

Clearly, unless genotypes g_i and g_k are equal, they cannot be explained by the same pair of haplotypes. Therefore, two different genotypes must be explained by at most one common haplotype. In practice, this constraint is integrated in the model by adding cardinality constraints on the variables x which capture the number of distinct haplotypes used to explain a pair of genotypes. Moreover, for incompatible pairs of genotypes, the constraint on the x variables is automatically guaranteed. Hence, for each pair of distinct non-homozygous compatible genotypes, at least three of their four pairwise haplotypes must be different:

$$\text{If } \kappa(i,k) \wedge g_i \neq g_k \wedge \exists_{j,j'}(g_{ij} = 2 \wedge g_{ij'} = 2), \text{ then } \sum_{p,q \in \{a,b\}} x_{i\,k}^{p\,q} \geq 3. \tag{2}$$

5 Experimental Results

A comprehensive evaluation was performed, using a set of 1183 problem instances (described in [3]), that include real and artificially generated problem instances. NRPoly has been compared against the other HIPP solvers. NRPoly uses the PBO solver MiniSat+ [2]. For the models using ILP, CPLEX version 11 was used. All HIPP solvers were run on a Intel Xeon 5160 server (3.0GHz, 1333Mhz, 4GB) running Red Hat Linux.

Figure 1 (left) provides a table with the number of aborted instances by NRPoly and the other HIPP algorithms, including the approaches in which NRPoly has been directly inspired: RTIP, SHIPs and RPoly. The total number of instances not solved within the time limit of 1000 seconds is given for each solver. We should note, however, that for RTIP many of the aborted instances exhausted the memory resources before the time limit. For SHIPs, the most recent version [7], which includes the lower bound used

Algorithms	# Aborted
NRPoly	**18**
RPoly	36
SHIPs	67
RTIP	378
Hapar	603
HybridIP	708
PolyIP	709

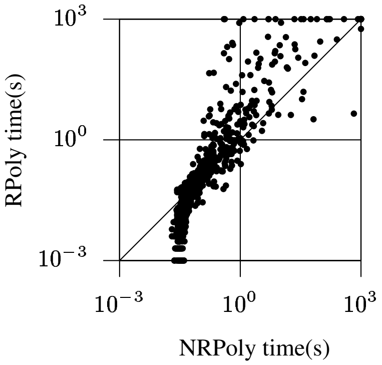

Fig. 1. Instances aborted by HIPP solvers within 1000s and performance of RPoly *vs* NRPoly.

by NRPoly, was considered. As can be concluded, the HIPP algorithms based on SAT or PBO are the most effective. NRPoly is the most robust algorithm aborting only 18 problem instances, thus reducing in half the number of instances aborted by RPoly.

Figure 1 (right) compares NRPoly with the best performing tool RPoly. For very easy instances RPoly is clearly faster (mainly due to the additional constraints of NRPoly) but for difficult instances NRPoly is consistently faster. There is only one exception for one problem instance that RPoly is able to solve a few seconds before the timeout and NRPoly is not. However, we have observed that NRPoly would be able to solve the same instance if it was allowed a few more seconds. Overall, we may conclude that NRPoly is more robust and more effective on solving the hardest instances.

Acknowledgments This work is partially funded by FCT 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. Integer programming approaches to haplotype inference by pure parsimony. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(2):141–154, 2006.
2. N. Eén and N. Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26, 2006.
3. A. Graça, J. Marques-Silva, I. Lynce, and A. Oliveira. Efficient haplotype inference with pseudo-Boolean optimization. In *Algebraic Biology (AB 2007)*, pages 125–139, 2007.
4. D. Gusfield. Haplotype inference by pure parsimony. In *14th Annual Symposium on Combinatorial Pattern Matching (CPM'03)*, pages 144–155, 2003.
5. G. Lancia, C. Pinotti, and R. Rizzi. Haplotyping populations by pure parsimony: complexity of exact and approximation algorithms. *INFORMS Journal on Computing*, 16:348–359, 2004.
6. I. Lynce and J. Marques-Silva. Efficient haplotype inference with Boolean satisfiability. In *National Conference on Artificial Intelligence (AAAI)*, 2006.
7. I. Lynce, J. Marques-Silva, and S. Prestwich. Boosting haplotype inference with local search. *Constraints*, 13(1), 2008.
8. L. Wang and Y. Xu. Haplotype inference by maximum parsimony. *Bioinformatics*, 19(14):1773–1780, 2003.