# Efficient Haplotype Inference with Boolean Satisfiability

**Inês Lynce**
IST/INESC-ID
Technical University of Lisbon, Portugal
ines@sat.inesc-id.pt

**João Marques-Silva**
School of Electronics and Computer Science
University of Southampton, UK
jpms@ecs.soton.ac.uk

## Abstract

One of the main topics of research in genomics is determining the relevance of mutations, described in haplotype data, as causes of some genetic diseases. However, due to technological limitations, genotype data rather than haplotype data is usually obtained. The haplotype inference by pure parsimony (HIPP) problem consists in inferring haplotypes from genotypes s.t. the number of required haplotypes is minimum. Previous approaches to the HIPP problem have focused on integer programming models and branch-and-bound algorithms. In contrast, this paper proposes the utilization of Boolean Satisfiability (SAT). The proposed solution entails a SAT model, a number of key pruning techniques, and an iterative algorithm that enumerates the possible solution values for the target optimization problem. Experimental results, obtained on a wide range of instances, demonstrate that the SAT-based approach can be several orders of magnitude faster than existing solutions. Besides being more efficient, the SAT-based approach is also the only capable of computing the solution for a large number of instances.

## Introduction

Over the last few years, an emphasis in human genomics has been on identifying genetic variations among populations. A comprehensive search for genetic influences on disease involves examining all genetic differences in a large number of affected individuals. This allows systematic testing of common genetic variants for their role in disease. The next high priority phase of human genomics will involve the development of a full Haplotype Map of the human genome. The HapMap Project (The International HapMap Consortium 2003) represents a key effort to develop a public resource that will help researchers to find genes associated with human disease. The achievement of this goal hinges on the ability for efficiently inferring haplotypes from genotypes.

There are two major approaches for solving the haplotype inference problem: combinatorial methods and statistical methods. Combinatorial methods often follow an optimization criterion (see (Gusfield & Orzach 2005)), whereas statistical methods usually follow a model of haplotype evolution (e.g. see (Stephens, Smith, & Donelly 2001)).

A well-known combinatorial approach to the haplotype inference problem is called Haplotype Inference by Pure Parsimony (HIPP). This problem is is APX-hard (Lancia, Pinotti, & Rizzi 2004) (and so NP-hard). The goal concerns finding the minimum number of distinct haplotypes for a given set of genotypes. Current approaches for solving the HIPP problem utilize Integer Linear Programming (ILP) (Gusfield 2003; Brown & Harrower 2004; 2005a) and branch and bound algorithms (Wang & Xu 2003). Observe that SAT has not been used in the past in the context of the HIPP problem, even though 2SAT has been used for solving a restricted form of haplotype inference, under the perfect phylogeny assumption (Halperin & Karp 2004).

The contribution of this paper is three-fold. First, we introduce a plain SAT model for encoding the haplotype inference by pure parsimony problem. Second, we introduce search pruning techniques for making the initial model practical for solving existing problem instances. These techniques include the elimination of symmetries by lexicographic ordering, the computation of lower bounds and the identification of structural properties of genotypes. Third, we provide experimental results that demonstrate the efficiency of the new approach.

## Basic Definitions

The DNA is a double-stranded molecule held together by weak bonds between base pairs of *nucleotides*. The position of a specific nucleotide is called a *site* or *locus*. There are four different types of nucleotides in DNA, which can be distinguished by the bases they contain. These bases are adenine (A), guanine (G), cytosine (C), and thymine (T). Base pairs are formed only between A and T and between G and C; thus the base sequence of each single strand can be deduced from that of the other strand in the DNA.

*Replication* is performed by first splitting the DNA double strand, and afterwards recreating each one of the two new strands with the corresponding bases. A *mutation* is an imperfection in the replication process, leading to DNA sequence variations: a base is accidentally skipped, inserted, or incorrectly copied. Once propagated to the next generation, a mutation may lead to variations within a population.

A *Single Nucleotide Polymorphism* or *SNP* is a DNA sequence variation, occurring when a single nucleotide is altered. For example, a SNP might change the nucleotide sequence AAGCCTA to AAGCTTA. SNPs make up 90% of all human genetic variations, and occur every 100 to 300 bases along the human genome. Variations in the DNA sequences of humans can affect how humans respond to diseases and treatments.

A *gene* is an ordered sequence of nucleotides located in a particular position that encodes a specific function. The variants of a single gene are named *alleles*. Different alleles give rise to differences in traits.

Different alleles may be explained in terms of SNPs. Depending on the number of possible alleles, a SNP site can be *biallelic* (two different alleles) or *multiallelic* (more than two different alleles). Almost always, there are only two possible alleles for a SNP site among the individuals in a population. In what follows we will only consider biallelic SNPs.

A *haplotype* is the genetic constitution of a sequence of nucleotides. The underlying data that forms a haplotype can be the full DNA sequence in the region, or more commonly the SNPs in that region. Diploid organisms pair homologous sequences, and thus contain two haplotypes, one inherited from each parent. The *genotype* describes the conflated data of the two haplotypes. In other words, an *explanation* for a genotype is a pair of haplotypes. If for a given site both copies of the haplotype have the same value, then the genotype is said to be *homozygous* at that site; otherwise is said to be *heterozygous*.

## Haplotype Inference

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 distinct), 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}, with homozygous sites being represented by values 0 or 1, depending on whether both haplotypes have value 0 or 1 at that site, respectively, and heterozygous sites being represented by value 2.

One of the approaches to the haplotype inference problem is called Haplotype Inference by Pure Parsimony (HIPP). A solution to this problem minimizes the total number of distinct haplotypes used. The HIPP problem is APX-hard (see (Gusfield 2003; Lancia, Pinotti, & Rizzi 2004) for proofs and historical perspective). Experimental results provide support for this approach (Wang & Xu 2003): the number of haplotypes in a large population is typically very small, although genotypes exhibit a great diversity. For example, consider the set of genotypes: 2120, 2102, and 1221. There are solutions for this example that use six distinct haplotypes, but the solution 0100/1110, 0100/1101, 1011/1101 uses only four distinct haplotypes.

## SAT-Based Haplotype Inference

The SAT-based formulation models whether there exists a set $\mathcal{H}$ of distinct haplotypes, with $r = |\mathcal{H}|$ haplotypes, such that each genotype $g_i \in \mathcal{G}$ is explained by a pair of haplotypes in $\mathcal{H}$. The SAT-based algorithm considers increasing sizes for $\mathcal{H}$, from a lower bound $lb$ to an upper bound $ub$. Trivial lower and upper bounds are, respectively, 1 and $2 \cdot n$. The algorithm terminates for a size of $\mathcal{H}$ for which there exist $r = |\mathcal{H}|$ haplotypes such that every genotype in $\mathcal{G}$ is explained by a pair of haplotypes in $\mathcal{H}$.

### The Plain SAT Model

In what follows we assume $n$ genotypes each with $m$ sites. The same indexes will be used throughout: $i$ ranges over the genotypes and $j$ over the sites, with $1 \leq i \leq n$ and $1 \leq j \leq m$. In addition, $r$ candidate haplotypes are considered, each with $m$ sites. An additional index $k$ is associated with haplotypes, with $1 \leq k \leq r$. As a result, $h_{kj} \in \{0, 1\}$ denotes the $j^{th}$ site of haplotype $k$. Moreover, a haplotype $h_k$, is viewed as a $m$-bit word $h_{k\,1} \dots h_{k\,m}$. A valuation $v : \{h_{k\,1}, \dots, h_{k\,m}\} \to \{0, 1\}$ to the bits of $h_k$ is denoted by $h_k^v$.

For a given value of $r$, the model considers $r$ haplotypes and seeks to associate two haplotypes (which can possibly represent the same haplotype) with each genotype $g_i$. As a result, 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 $s_{ki}^a$ and $s_{ki}^b$. Hence, genotype $g_i$ is explained by haplotypes $h_{k_1}$ and $h_{k_2}$ if $s_{k_1 i}^a = 1$ and $s_{k_2 i}^b = 1$. Clearly, $g_i$ is also explained by the same haplotypes if $s_{k_2 i}^a = 1$ and $s_{k_1 i}^b = 1$.

If a site $g_{ij}$ of a genotype $g_i$ is 0 or 1, then this is the value required at this site and is used by the model. If a site $g_{ij}$ is 0, then the model requires, for $k = 1, \dots, r$:

$$(\neg h_{kj} \vee \neg s_{ki}^a) \wedge (\neg h_{kj} \vee \neg s_{ki}^b) \qquad (1)$$

Hence, if haplotype $k$ is selected for explaining genotype $i$, either by the $a$ or $b$ representatives, then the value of haplotype $k$ at site $j$ *must* be 0.

If a site $g_{ij}$ is 1, then the model requires, for $k = 1, \dots, r$:

$$(h_{kj} \vee \neg s_{ki}^a) \wedge (h_{kj} \vee \neg s_{ki}^b) \qquad (2)$$

Hence, if haplotype $k$ is selected for explaining genotype $i$, either by the $a$ or $b$ representatives, then the value of haplotype $k$ at site $j$ *must* be 1.

Otherwise, if a site $g_{ij}$ is 2, one requires that the haplotypes explaining the genotype $g_i$ have opposing values at site $j$. This is done by creating two variables, $g_{ij}^a, g_{ij}^b \in \{0, 1\}$, such that $g_{ij}^a \neq g_{ij}^b$. In CNF, the model requires two clauses:

$$(g_{ij}^a \vee g_{ij}^b) \wedge (\neg g_{ij}^a \vee \neg g_{ij}^b) \qquad (3)$$

As a result, for the case where a site $g_{ij}$ is 2, then the model requires, for $k = 1, \ldots, r$:

$$\begin{aligned}(h_{kj} \vee \neg g^a_{ij} \vee \neg s^a_{ki}) \wedge (\neg h_{kj} \vee g^a_{ij} \vee \neg s^a_{ki}) \wedge \\ (h_{kj} \vee \neg g^b_{ij} \vee \neg s^b_{ki}) \wedge (\neg h_{kj} \vee g^b_{ij} \vee \neg s^b_{ki})\end{aligned} \tag{4}$$

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

$$\left(\sum_{k=1}^{r} s^a_{ki} = 1\right) \wedge \left(\sum_{k=1}^{r} s^b_{ki} = 1\right) \tag{5}$$

Since the proposed model is purely SAT-based, a simple alternative solution is used, which requires the utilization of additional variables $v^a_{ki}$ and $v^b_{ki}$. The alternative solution consists of the CNF representation of a simplified adder circuit. We consider the case for the $a$ variables; for the $b$ variables the model is exactly the same. The $v^a_{ki}$ variables are defined as follows:

$$\begin{array}{l} v^a_{1i} \leftrightarrow s^a_{1i} \\ (\neg v^a_{ki} \vee \neg s^a_{ki}) \\ v^a_{k+1\,i} = (s^a_{ki} \vee v^a_{ki}) \\ v^a_{ri} = 1 \end{array} \tag{6}$$

Finally, and given that $1 \leq i \leq n$, $1 \leq j \leq m$ and $1 \leq k \leq r$, the number of variables and constraints in the proposed SAT model is, respectively, $\mathcal{O}(r_f\, m + r_f\, n + n\, m)$ and $\mathcal{O}(r_f\, n\, m)$, where $r_f$ is the final value of $r$. Since $r_f = \mathcal{O}(n)$, the number of variables and constraints becomes, respectively, $\mathcal{O}(n\, m + n^2)$ and $\mathcal{O}(n^2\, m)$, which is also the number of variables and constraints in the model proposed in (Brown & Harrower 2004). Nevertheless, our experience is that $r_f$ is in general *much smaller* than $n$, and so our model yields significantly more compact representations than the models of (Brown & Harrower 2004; 2005a).

It is important to observe that the model proposed above is not practical for most of the existing problem instances, even with the most efficient SAT solvers. As a result, a number of techniques has been developed with the goal of providing significant search pruning.

## Breaking Symmetries

A key technique for pruning the search space is motivated by observing the existence of symmetry in the problem formulation.

Consider two haplotypes $h_{k_1}$ and $h_{k_2}$, and the selector variables $s^a_{k_1 i}$, $s^a_{k_2 i}$, $s^b_{k_1 i}$ and $s^b_{k_2 i}$. Furthermore, consider Boolean valuations $v_x$ and $v_y$ to the sites of haplotypes $h_{k_1}$ and $h_{k_2}$. Then, $h^{v_x}_{k_1}$ and $h^{v_y}_{k_2}$, with $s^a_{k_1 i} s^a_{k_2 i} s^b_{k_1 i} s^b_{k_2 i} = 1001$, corresponds to $h^{v_y}_{k_1}$ and $h^{v_x}_{k_2}$, with $s^a_{k_1 i} s^a_{k_2 i} s^b_{k_1 i} s^b_{k_2 i} = 0110$, and one of the assignments can be eliminated. To remedy this, one possibility is to enforce an ordering of the valuations to the haplotypes[1]. Hence, for any valuation $v$ to the problem variables we require $h^v_1 < h^v_2 < \ldots < h^v_r$.

It is straightforward to enforce each sorting constraint in linear size on the number of haplotypes and sites. This is done by representing in CNF a Boolean comparator circuit between $h_k$ and $h_{k+1}$, with $1 \leq k < r$, and requiring $h_k < h_{k+1}$.

## Computing Lower Bounds

Lower bounds allow reducing the number of iterations of the algorithm proposed in the previous section. Moreover, lower bounds also allow *reducing* the size of the model described in the previous sections.

Two genotypes, $g_i$ and $g_l$, are declared *incompatible* iff there exists a site for which the value of one genotype is 0 and the other is 1. For example, $g_1 = 012$ is incompatible with $g_2 = 112$, whereas the genotypes $g_1$ and $g_3 = 210$ are not incompatible. 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 have an edge iff they are incompatible. Suppose $I$ has a clique of size $k$, then the number of required haplotypes is at least $2 \cdot k - \sigma$, where $\sigma$ is the number of genotypes in the clique which do not have heterozygous sites. In order to find the largest lower bound, our objective is to identify the maximum clique in $I$. Since this problem is NP-hard, we settle with computing a maximal clique in the incompatibility graph. Currently, a greedy procedure is used, which starts from the genotype with the largest number of incompatible genotypes and at each step adds genotypes incompatible with all genotypes included in the clique. Given a lower bound $lb$, the algorithm described in the previous section is only required to enumerate tentative sizes for the number of haplotypes between $lb$ and $2 \cdot n$.

Moreover, we note that the information regarding the lower bound can be used for *reducing* the size of the model. If a genotype $g_i$ is part of the clique and has at least one heterozygous site, then we can associate two dedicated haplotypes with $g_i$. If a genotype $g_i$ is part of the clique and all its sites are homozygous, then we associate only one dedicated haplotype with $g_i$. In addition, when considering the candidate haplotypes for a genotype $g_l$, which is incompatible with genotype $g_i$ included in the clique, the haplotypes associated with $g_i$ need not be considered as candidates for $g_l$. This eliminates $s$ variables and the corresponding clauses.

We conclude this section by noting that it is possible to increase the lower bound obtained with a maximal clique. Suppose a genotype $g_i$, not part of the maximal clique, which is heterozygous at site $j$, and further assume that all other genotypes assume the same homozygous value (either 0 or 1) at site $j$. Then, it is straightforward to conclude that explaining genotype $g_i$ requires one haplotype which cannot be used to explain *any* of the other genotypes. Hence, $g_i$ can be used to increase the lower bound by 1.

## Identifying Structural Properties

One additional simplification consists of using the structural properties of genotypes with the purpose of reducing the search space. These simplifications have been used by others (Brown & Harrower 2005b).

[1] See for example (Frisch *et al.* 2002) for a survey on the utilization of lexicographic orderings for symmetry breaking.

The first observation is that sets of genotypes often include duplicate genotypes. Clearly, in the presence of two equal genotypes, one can be discarded, assuming the two genotypes are explained identically. Hence, the solution for the remaining genotype is also the solution for the discarded genotype. Another observation is that duplicate sites can be discarded, i.e. sites for which each genotype has equal values. Finally, complemented sites can also be discarded, where two sites are *complemented* iff the homozygous sites have complemented values.

## Experimental Results

The main goal of this section is to demonstrate that the HIPP problem is effectively solved using our model and a state-of-the art SAT solver. We have implemented the algorithm described in the previous section using a Perl script that encodes the problem to be given to the minisat SAT solver (Eén & Sörensson 2003). Our solution is called SHIPs (Sat-based Haplotype Inference by Pure Parsimony).

For evaluating SHIPs, as well as other available tools for solving the HIPP problem, we have collected a significant number of problem instances. Problem instances may be obtained following two different approaches: generating problem instances or obtaining real problem instances. Problem instances are typically generated using Hudson's program ms (Hudson 2002). This program generates *haplotypes* following a standard coalescent approach in which the genealogy of the sample is first randomly generated and then mutations are randomly placed on the genealogy. Given the haplotypes, genotypes are generated and given to a HIPP solver. In addition, the HapMap project (http://www.hapmap.org/) provides a comprehensive source of *genotype* data over four populations.

Experiments performed by Gusfield (Gusfield 2003) demonstrate that the HIPP approach is particularly accurate on solving problems with moderate levels of recombination. (Recombination occurs whenever a haplotype results from the combination between the two haplotypes of a parent.) For these problems, 80% to 95% of the inferred haplotypes are correct. Moreover, an extensive experimental evaluation presented in (Wang & Xu 2003) demonstrates that the HIPP approach is competitive with other approaches, performing different forms of haplotype inference (Stephens, Smith, & Donelly 2001). Furthermore, the HIPP approach is competitive in terms of accuracy, and significantly faster in terms of performance.

For our experiments we have used three sets of benchmarks. These benchmarks are obtained as follows:

1. **Uniform:** Given a set of haplotypes obtained by using the ms program, remove repeated haplotypes. Randomly pick any two haplotypes as an explanation for a genotype.
2. **Non-uniform:** Given a set of haplotypes obtained by using the ms program, pairs are obtained by randomly pairing two haplotypes, which form a genotype. Repeated haplotypes are not removed and so have a higher probability of being picked.
3. **Hapmap:** Genotype inputs obtained over all four HapMap populations: Yoruba of Ibadan - Nigeria, Japanese of Tokyo, Chinese of Beijing and US with northern and western European ancestry. For each DNA sequence, a continuous collection of SNPs with a small amount of recombination was selected.

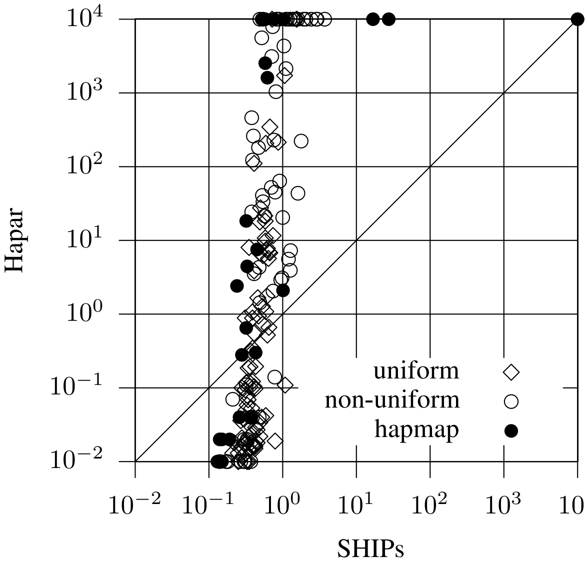

Figure 1: Hapar vs SHIPs on uniform, non-uniform and hapmap instances.

Table 1 provides results for a set of 229 problem instances, including uniform, non-uniform and hapmap instances [2]. The ms program has been used to generate uniform and non-uniform instances with 10, 30, 50, 75 and 100 sites. Uniform instances with 10 sites include different recombination levels: 0, 4 and 16. All the other instances have recombination level 0. Moreover, instances with 10 and 30 sites have 50 genotypes, whereas instances with 50, 75 and 100 sites have only 30 genotypes. For the problems obtained from the HapMap project, sequences of lengths 30, 50 and 75 were tested. The number of genotypes in these problems range from 7 to 68 genotypes.

Table 1 compares the performance of RTIP (Gusfield 2003), Poly (Brown & Harrower 2004), Hybrid (Brown & Harrower 2005a), Hapar (Wang & Xu 2003) and SHIPs on solving these 229 problem instances. The first three solvers were provided by D. Brown and I. Harrower and Hapar was provided by its authors. For each solver is given the number of problems solved within 1000s using a 1.9 GHz AMD Athlon XP with 1GB of RAM running RedHat Linux.

From Table 1 it is clear that Hapar and SHIPs are by far the most effective solvers. All others abort on many more problem instances. Moreover, SHIPs aborts 1 single instance out of 229, in contrast with Hapar which aborts 39 out of 229.

Figure 1 compares the CPU time required by both Hapar and SHIPs on solving each of the 229 problem instances. (A log scale has been used.) Observe that for this plot the limit CPU time was extended to 10000s. With the only ex-

[2] Instances provided by D. Brown and I. Harrower.

Table 1: Results for RTIP, Poly and Hybrid, Hapar and SHIPs.

| Benchmarks | Sites | Recomb | Genotypes | RTIP | Poly | Hybrid | Hapar | **SHIPs** |
|---|---|---|---|---|---|---|---|---|
| Uniform | 10 | – | 50 | 15/15 | 15/15 | 15/15 | 15/15 | 15/15 |
| | 10 | 4 | 50 | 15/15 | 14/15 | 14/15 | 15/15 | 15/15 |
| | 10 | 16 | 50 | 15/15 | 6/15 | 6/15 | 15/15 | 15/15 |
| | 30 | – | 50 | 6/15 | 4/15 | 3/15 | 15/15 | 15/15 |
| | 50 | – | 30 | 0/50 | 12/50 | 13/50 | 50/50 | 50/50 |
| | 75 | – | 30 | 0/10 | 2/10 | 2/10 | 8/10 | 10/10 |
| | 100 | – | 30 | 0/10 | 0/10 | 1/10 | 9/10 | 10/10 |
| Non-Uniform | 10 | – | 50 | 15/15 | 14/15 | 14/15 | 15/15 | 15/15 |
| | 30 | – | 50 | 11/15 | 1/15 | 2/15 | 15/15 | 15/15 |
| | 50 | – | 30 | 3/15 | 0/15 | 1/15 | 12/15 | 15/15 |
| | 75 | – | 30 | 2/15 | 0/15 | 0/15 | 4/15 | 15/15 |
| | 100 | – | 30 | 1/15 | 0/15 | 0/15 | 4/15 | 15/15 |
| Hapmap | 30:75 | – | 7:68 | 0/24 | 12/24 | 12/24 | 13/24 | 23/24 |
| **TOTAL** | 10:100 | 0:16 | 7:68 | 83/229 | 80/229 | 83/229 | 190/229 | 228/229 |

ception of a hapmap problem instance for which both SHIPs and Hapar abort, each of the problem instances was solved by SHIPs in less than 30s. Although within 10000s Hapar aborted on 20 instances, around 80% of the instances were solved in less than 100s. Moreover, and besides trivial instances (for which both SHIPs and Hapar take less than 1s), SHIPs is in general faster than Hapar by 1 to 4 orders of magnitude.

In order to further compare the performance of SHIPs and Hapar, we have performed experiments on more 175 hard problem instances. Some of the instances were also provided by Brown and Harrower: 70 uniform instances with 50 genotypes each (15 instances with 10 sites and recombination level 40, 3 $\times$ 15 instances with 30 sites and recombination levels 4, 16 and 40 and 10 instances with 50 sites) and 15 non-uniform instances (with 50 sites and 50 genotypes). Furthermore, we have generated more 3 $\times$ 15 uniform and 3 $\times$ 15 non-uniform problem instances with 75 sites and 50 genotypes, 100 sites and 50 genotypes, and 100 sites and 100 genotypes.

Figure 2 provides a plot comparing the CPU time required by Hapar and SHIPs for solving each of the additional 175 problem instances within 10000s. This comparison between Hapar and SHIPs clearly demonstrates the supremacy of the new SAT-based approach. Similarly to the previous results, and besides trivial instances, SHIPs is in general between 2 and 4 orders of magnitude faster than Hapar. In addition, SHIPs aborted only 2 problem instances out of 175 ($\sim$ 1%), while Hapar aborted 79 problem instances ($\sim$ 45%), including most of the non-uniform instances.

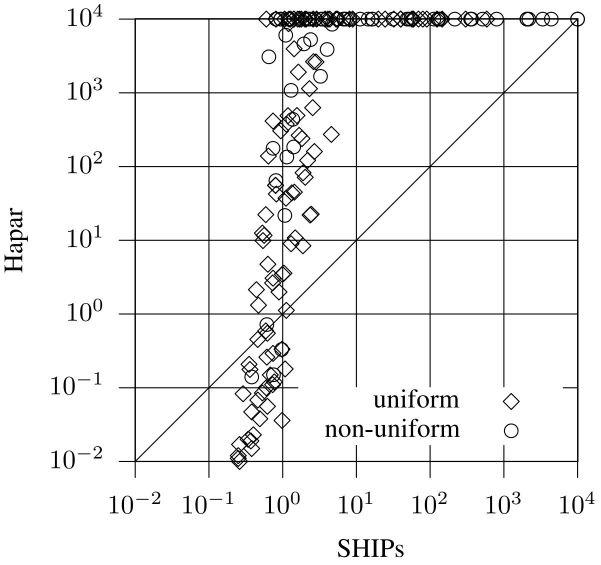

Figure 2: Hapar vs SHIPs on additional uniform and non-uniform instances.

## Related Work

Over the last few years, a number of authors have proposed solutions for the HIPP problem (Gusfield & Orzach 2005). With a few notable exceptions (Wang & Xu 2003), the majority of the proposed solutions are based on Integer Linear Programming (ILP) (Gusfield 2003; Brown & Harrower 2004; 2005a). The original ILP model, *RTIP*, has linear space complexity on the number of possible haplotypes (Gusfield 2003), and so exponential on the number of given genotypes. A Boolean variable $y_r$ is associated with each pair of haplotypes that can explain a given genotype $g_i$, and denotes whether these pair of haplotypes is used for explaining $g_i$. A cardinality constraint 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 $x_s$, which denotes whether the haplotype is used. The utilization of a specific pair of haplotypes for explaining a genotype implies the respective $x_s$ variable. The cost function consists of minimizing the number of $x_s$ variables assigned value 1. Techniques for eliminating irrelevant pairs of haplotypes have been developed (Gusfield 2003). In (Wang & Xu 2003), the RTIP model was adapted to a branch and bound algorithm, *Hapar*. In addition, key reductions on the size of the model are achieved by identifying haplotypes that may only explain one or two genotypes. A more recent ILP model, *Poly*, is polynomial in the number of sites and population size (Brown & Harrower 2004), with a number of constraints in $\Theta(n^2m)$. More recently, (Brown & Harrower 2005a) introduced a new polynomial-size formulation, *Hybrid*, consisting of a hybrid of the two ILP formulations above.

## Conclusions and Future Work

This paper proposes SHIPs, the first SAT-based approach for the problem of inferring haplotypes under the pure parsimony criterion.

The results presented in the paper are conclusive. The SHIPs approach is much more efficient than all existing combinatorial approaches, either based on integer linear programming or on dedicated branch-and-bound solutions. Besides being in general several orders of magnitude faster than the previous best existing solution (i.e. Hapar) for non-trivial instances, SHIPs is also capable of solving a large number of instances that no other approach is capable of.

Despite SHIPs being extremely efficient on existing problem instances, and even on significantly more complex problem instances specifically generated for this paper, several challenges still remain. The practical utilization of SHIPs on significantly larger real-world instances will create hard problem instances, which current SAT solver technology may be unable to tackle. Expected improvements include the development of additional pruning techniques, more sophisticated lower bounding, and a more optimized encoding into SAT.

## Acknowledgments

The authors are very grateful to Arlindo Oliveira for having pointed out the HIPP problem. This work is partially supported by Fundação para a Ciência e Tecnologia under research project POSC/EIA/61852/2004.

## References

Brown, D., and Harrower, I. 2004. A new integer programming formulation for the pure parsimony problem in haplotype analysis. In *Proceedings of the 4th Workshop on Algorithms in Bioinformatics (WABI'04)*.

Brown, D., and Harrower, I. 2005a. A new formulation for haplotype inference by pure parsimony. Technical Report CS-2005-03, University of Waterloo, School of Computer Science.

Brown, D., and Harrower, I. 2005b. Personal communication.

Eén, N., and Sörensson, N. 2003. An extensible satsolver. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT'03)*, 502–518.

Frisch, A.; Hnich, B.; Kiziltan, Z.; Miguel, I.; and Walsh, T. 2002. Global constraints for lexicographic orderings. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming (CP'02)*.

Gusfield, D., and Orzach, S. 2005. *Handbook on Computational Molecular Biology*, volume 9 of *Chapman and Hall/CRC Computer and Information Science Series*. CRC Press. chapter Haplotype Inference.

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

Halperin, E., and Karp, R. M. 2004. Perfect phylogeny and haplotype assignment. In *Proceedings of the 8th Annual International Conference on Computational Molecular Biology (RECOMB'04)*, 10–19.

Hudson, R. R. 2002. Generating samples under a wright-fisher neutral model of genetic variation. *Bioinformatics* 18(2):337–338.

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

Stephens, M.; Smith, N.; and Donelly, P. 2001. A new statistical method for haplotype reconstruction. *American Journal of Human Genetics* 68:978–989.

The International HapMap Consortium. 2003. The international hapmap project. *Nature* 426:789–796.

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