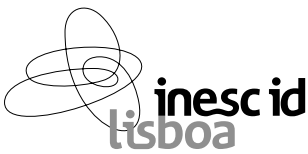
Assessing the efficacy of haplotype inference by pure parsimony on biological data

Inês Lynce, João P. Marques-Silva and Arlindo L. Oliveira

Technical Report 12/2006

March 2006



Software Algorithms and Tools for constraint solving - SAT

Instituto de Engenharia de Sistemas e Computadores, Investigação e Desenvolvimento, Lisboa

Abstract

Motivation: Mutation in DNA is the principal factor responsible for the differences among human beings, and Single Nucleotide Polymorphisms (SNPs) are the most common mutations. Hence, it is fundamental to complete a map of haplotypes (which identify SNPs) in the human population. 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. Experimental results provide support for this method: the number of haplotypes in a large population is typically very small, although genotypes exhibit a great diversity.

Results: We define a new method for solving the HIPP problem using Boolean satisfiability (SAT). Experimental results clearly demonstrate that our SAT-based tool is by far more efficient than the existing tools following the pure parsimony approach. Besides being more efficient, our tool is also the only capable of computing the solution for a large number of instances. Hence, we have been able to infer haplotypes for a wide variety of biological data, that no other tool following the pure parsimony approach is able to. Additionally, we compare the accuracy of the results obtained with our tool with the accuracy of the results obtained with tools following statistical approaches, and conclude that the pure parsimony criterion used by our methods leads to good results in terms of precision of inferred haplotypes.

1 Introduction

Over the last few years, an emphasis in human genomics has been on identifying genetic variations among different people. A comprehensive search for genetic influences on disease involves examining all genetic differences in a large number of affected individuals. This allows to systematically test common genetic variants for their role in disease; such variants explain much of the genetic diversity in our species, a consequence of the historically small size and shared ancestry of the human population.

A particular focus has been put on the identification of Single Nucleotide Polymorphisms (SNPs), point mutations found with only two common values in the population, and tracking their inheritance. However, this process is in practice very difficult due to technological limitations. At a genomic position for which an individual inherited two different values, it is currently difficult to identify from which parent each value was inherited. Instead, researchers can only identify that the individual is heterozygotic at that position, i.e. that the values inherited from both parents are different. If we could identify maternal and paternal inheritance better, we could trace the structure of the human population more accurately and improve our ability to map disease genes. This process of going from genotypes (which include the ambiguity at heterozygous positions) to haplotypes (where we know from which parent each SNP is inherited) is called *haplotype inference*.

The next high priority phase of human genomics will involve the development of a full Haplotype Map of the human genome. The HapMap Project [25] represents the best known effort of scientists to develop a public resource that will help researchers to find genes associated with human disease. The HapMap resource can guide the design and analysis of genetic association studies, shed light

on structural variation and recombination, and identify sites that may have been subject to natural selection during human evolution. The achievement of these goals depends on an efficient method that performs inference of haplotypes from genotypes.

A well-known approach to the haplotype inference problem is called Haplotype Inference by Pure Parsimony (HIPP). The problem of finding such solutions is APX-hard (and, therefore, NP-hard) [16]. The Pure-Parsimony problem is to find a solution to the haplotype inference problem that minimizes the total number of distinct haplotypes used. Current methods for solving the haplotype inference problem by pure parsimony utilize Integer Linear Programming (ILP) [11, 2, 4] and branch and bound algorithms [26].

Other existing approaches are based on statistical methods (e.g. see [24, 21]). These methods start by inferring haplotype frequencies and afterwards use these frequencies to compute the haplotypes that explain the genotypes. The problem of finding such solutions can be made polynomial.

Although experiments results given in [11, 26] indicate that the accuracy of the HIPP approach is competitive when compared with other approaches, most of the tools for solving the HIPP problem are known for solving only very small problem instances. Indeed, this fact has motivated the use of approximation algorithms inspired on pure parsimony [13]. Nonetheless, we argue that despite the worst-case exponential case of all known algorithms for solving the HIPP problem, it may be possible to build efficient tools. This is actually the case with Boolean Satisfiability (SAT) [1] [20, 9], where state-of-the-art SAT solvers are currently able of solving formulas with hundreds of thousand variables and tens of million clauses.

The contribution of this paper is three-fold. First, we describe a plain SAT model for encoding the haplotype inference by pure parsimony problem. Second, we provide experimental results that demonstrate the efficiency of the new model when compared with other models for solving the HIPP problem. Finally, we provide experimental results that demonstrate that the HIPP approach is as accurate as statistical-based approaches.

This paper is organized as follows. The next section describes the haplotype inference problem and the pure parsimony approach. Afterwards, we describe Boolean satisfiability, which is the basis for our model for solving the HIPP problem. Section 4 explains how to compute haplotypes using Boolean satisfiability, including a description of the SAT model. Next, we give experimental results that, besides demonstrating that our tool is much more efficient than existing tools for haplotype inference by pure parsimony, also compare the accuracy of the HIPP approach with the accuracy of other approaches on a wide range of biological data. This extensive comparison shows that the HIPP approach is as accurate as other well-known approaches. Finally, the paper concludes and suggests future research work.

[1] SAT was the first problem to be proved to be NP-complete [5].

2 Haplotype Inference by Pure Parsimony

A *haplotype* is the genetic constitution of an individual chromosome. 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 chromosomes, 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. Conversely, this pair of haplotypes explains the genotype. 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*.

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, such that for each genotype $g_i \in G$ 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 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.

Table 1 gives twelve haplotypes of the β_2AR genes [2]. Each haplotype has 13 sites. Each site corresponds to a specific nucleotide in a gene where a mutation occurred. Each nucleotide is characterized by the position in the sequence. For each nucleotide, a pair of possible alleles is given: the first allele corresponds to the wild type and the second to the mutant. The last column of each haplotype contains the representation of that haplotype. Different genotypes may result from this set of haplotypes: for example, the pair of haplotypes (h_1, h_7) explains the genotype 200000020222, whereas the pair (h_7, h_8) explains the genotype 0020000020121.

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 [11, 16] for proofs and historical perspective). Experimental results provide support for this method: 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.

Over the last few years, a number of authors have proposed optimization models for the HIPP problem, most of which based on ILP [12]. With a few notable exceptions [26], the majority of the

[2] These data were obtained from [8].

Table 1: Haplotypes of the β_2AR genes.

Nucleotide	-1023	-709	-654	-468	-406	-367	-47	-20	46	79	252	491	523	
Alleles	G/A	C/A	G/A	C/G	C/T	T/C	T/C	T/C	G/A	C/G	G/A	C/T	C/A	
h_1	A	C	G	C	C	T	T	T	A	C	G	C	C	1000000010000
h_2	A	C	G	G	C	C	C	C	G	G	G	C	C	1001011101000
h_3	G	A	A	C	C	T	T	T	A	C	G	C	C	0110000010000
h_4	G	C	A	C	C	T	T	T	A	C	G	C	C	0010000010000
h_5	G	C	A	C	C	T	T	T	G	C	G	C	C	0010000000000
h_6	G	C	G	C	C	T	T	T	G	C	A	C	A	0000000000101
h_7	G	C	G	C	C	T	T	T	G	C	A	T	A	0000000000111
h_8	G	C	A	C	C	T	T	T	A	C	A	C	A	0010000010101
h_9	A	C	G	C	T	T	T	T	A	C	G	C	C	1000100010000
h_{10}	G	C	G	C	C	T	T	T	G	C	A	C	C	0000000000100
h_{11}	G	C	G	C	C	T	T	T	G	C	G	C	C	0000000000000
h_{12}	A	C	G	G	C	T	T	T	A	C	G	C	C	1001000010000

proposed models utilize Integer Linear Programming (ILP) [11, 2, 4].

The original ILP models are linear in the number of possible haplotypes [11], and so exponential on the number of given genotypes. A new variable is created for each pair of haplotypes that can be used for explaining a given genotype. The objective function is to minimize the number of variables representing distinct haplotypes. This formulation is commonly referred to as RTIP.

In [26], the RTIP model was adapted to a branch and bound algorithm that searches all the possible solutions and returns the one with the fewer number of haplotypes. In addition, remarkable reductions on the size of the model are achieved by identifying haplotypes that may only explain one or two genotypes. This formulation was implemented in the Hapar solver.

Recent ILP models are polynomial in the number of sites and population size [2]. For each site in each genotype, two binary variables are created. The sum of each pair of variables depends on the value of the genotype in the corresponding site. This formulation is commonly referred to as Poly. The size of the Poly model is $\mathcal{O}(n^2m)$.

More recently, Brown and Harrower [4] introduced a new polynomial-size formulation that is a hybrid of the two ILP formulations above and inherits many of the strengths of both. In addition to the variables mentioned above, a small set of possible explaining haplotypes is created. The final solution may include some of these haplotypes. This formulation is commonly referred to as Hybrid.

3 Boolean Satisfiability

The Boolean satisfiability (SAT) problem is represented using n propositional variables $x_1, x_2, \ldots, x_n$, which can be assigned truth values 0 (false) or 1 (true). A literal l is either a variable x_i (i.e., a positive literal) or its complement $\neg x_i$ (i.e., a negative literal). A clause ω is a disjunction ($\vee$) of

literals and a CNF formula φ is a set of clauses (interpreted as a conjunction ($\wedge$) of clauses). A literal l_j of a clause ω_a that is assigned truth value 1 satisfies the clause, and the clause is said to be *satisfied*. If the literal is assigned truth value 0 then it is removed from the clause. A clause with a single literal is said to be *unit* and its literal has to be assigned value 1 for the clause to be satisfied. The iterative application of this rule is called unit propagation. The derivation of an *empty* clause indicates a conflict and therefore the formula is *unsatisfied* for the given assignment. The formula is *satisfied* if all its clauses are satisfied. The SAT problem consists of deciding whether there exists a truth assignment to the variables such that the formula becomes satisfied.

For example, consider formula $\varphi = \{\omega_1, \omega_2\}$, with $\omega_1 = \{x_1\}$ and $\omega_2 = \{\neg x_1 \vee x_2\}$. Simply by applying unit propagation we get the satisfying assignment $x_1 = 1$ and $x_2 = 1$. Now consider formula $\varphi' = \varphi \cup \{\omega_3\}$ with $\omega_3 = \{\neg x_1 \vee \neg x_2\}$. Clearly, there is no truth assignment such that φ' becomes satisfied.

Over the years a large number of algorithms has been proposed for SAT, from the original Davis-Putnam procedure [7], followed by Davis-Logemann-Loveland backtrack search procedure (DLL) [6], to recent backtrack search algorithms and to local search algorithms, among many others. Local search algorithms can solve extremely large satisfiable instances of SAT. These algorithms have also been shown to be very efficient on randomly generated instances of SAT. On the other hand, several improvements to the DLL backtrack search algorithm have been introduced. These improvements have been shown to be crucial for solving large instances of SAT derived from real-world applications, and in particular for those where local search cannot be applied, i.e. for unsatisfiable instances. Indeed, proving unsatisfiability is often the objective in a large number of significant real-world applications.

Recent state-of-the-art SAT solvers utilize different forms of intelligent backtracking [1, 19, 27]. In these algorithms each identified conflict is analyzed to identify the variable assignments that caused it, and a new clause (*nogood*) is created to explain and prevent the identified conflicting conditions from happening again. The created clause is then used to compute the backtrack point. Moreover, some of the recorded clauses are eventually deleted.

Efficient implementations represent the most recent paradigm shift in SAT solvers. Due to the huge size of many of the benchmark problem instances, a careful implementation can make the difference between being able or unable to solve a given problem instance in a reasonable amount of time. Moreover, learning new clauses may significantly increase the clause database size, though this technique is essential for solving hard real-world instances. This motivates the use of very efficient data structures. In recent years, they have evolved from intuitive to more sophisticated data structures. Efficient implementations for backtrack search SAT solvers with learning are based on lazy data structures [20]. The laziness in these structures allows the status of a clause to be imprecisely known. Nevertheless, unit and empty clauses are always identified, ensuring that unit propagation is always applied and conflicts are always detected.

4 Computing Haplotypes with Boolean Satisfiability

This section reviews the SAT-based model introduced in [18, 17].

The SAT-based formulation models whether there exists a set $\mathcal{H}$ of 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 exists $r = |\mathcal{H}|$ haplotypes such that every genotype in $\mathcal{G}$ is explained by a pair of haplotypes in $\mathcal{H}$.

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, $1 \leq k \leq r$. As a result, $h_{kj} \in \{0, 1\}$ denotes the jth site of haplotype k. Moreover, a haplotype h_k, is viewed as a m-bit word, $h_{k\,1} \ldots h_{k\,m}$. A valuation $v : \{h_{k\,1}, \ldots, h_{k\,m}\} \rightarrow \{0, 1\}^m$ 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, $1 \leq i \leq n$. 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, with $k = 1, \ldots, r$. Hence, genotype g_i is explained by haplotypes h_{k_1} and h_{k_2} iff $s_{k_1 i}^a = 1$ and $s_{k_2 i}^b = 1$. Clearly, g_i is also explained by the same haplotypes iff $s_{k_2 i}^a = 1$ 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 $k = 1, \ldots, r$:

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

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

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

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

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

In addition, the model requires, for $k = 1, \ldots, r$:

$$\begin{array}{l} (h_{kj} \vee \neg g_{ij}^a \vee \neg s_{ki}^a) \wedge (\neg h_{kj} \vee g_{ij}^a \vee \neg s_{ki}^a) \wedge \\ (h_{kj} \vee \neg g_{ij}^b \vee \neg s_{ki}^b) \wedge (\neg h_{kj} \vee g_{ij}^b \vee \neg s_{ki}^b) \end{array} \tag{4}$$

Clearly, for each i, and for 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:

$$\left(\sum_{k=1}^{r} s_{ki}^{a} = 1\right) \wedge \left(\sum_{k=1}^{r} s_{ki}^{b} = 1\right) \tag{5}$$

Since the proposed model is purely SAT-based, a simple alternative solution is used, which consists of the CNF representation of a simplified adder circuit.

The complexity of the proposed SAT model becomes $\mathcal{O}(r\,n\,m)$. Since $r = \mathcal{O}(n)$, the proposed model is $\mathcal{O}(n^2\,m)$, which is the complexity of the model proposed in [2]. Nevertheless, our experience is that r is in general much smaller than n, and so our model yields significantly more compact representations than the models of [2, 4].

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 $s_{k_1 i}^{a}$, $s_{k_2 i}^{a}$, $s_{k_1 i}^{b}$ and $s_{k_2 i}^{b}$. Furthermore, consider Boolean valuations v_x and v_y to the sites of haplotypes h_{k_1} and h_{k_2}. Then, $h_{k_1}^{v_x}$ and $h_{k_2}^{v_y}$, with $s_{k_1 i}^{a} s_{k_2 i}^{a} s_{k_1 i}^{b} s_{k_2 i}^{b} = 1001$, corresponds to $h_{k_1}^{v_y}$ and $h_{k_2}^{v_x}$, with $s_{k_1 i}^{a} s_{k_2 i}^{a} s_{k_1 i}^{b} s_{k_2 i}^{b} = 0110$, and one of the assignments can be eliminated. To remedy this, one possibility is to enforce an ordering of the Boolean valuations to the haplotypes[3]. Hence, for any valuation v to the problem variables we require $h_1^v < h_2^v < \ldots < h_r^v$.

Another key technique for pruning the search space is the use of a lower bound on the size of the number of required haplotypes.

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 vertices have an edge if 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 σ 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 use the size of a maximal clique in the incompatibility graph, computed using a simple greedy heuristic.

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 haplotypes 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.

[3]See for example [10] for a survey of earlier work on the utilization of lexicographic orderings for symmetry breaking.

Furthermore, it is possible to increase the lower bound obtained with a maximal clique. Suppose a genotype g_i 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.

One additional simplification consists of using the structural properties of genotypes with the purpose of reducing the search space [4]. Clearly, in the presence of two equal genotypes, one can be discarded, assuming the two genotypes are phased identically. Hence, the solution for the remaining genotype is also the solution for the discarded genotype. Duplicate sites can also be discarded, i.e. sites for which each genotype has equal values. Moreover, complemented sites can also be discarded, where two sites are *complemented* if the homozygous sites have complemented values.

5 Experimental Results

The main goal of this section is two-fold: (1) to show that the HIPP problem is effectively solved using our model and a state-of-the art SAT solver and (2) to compare the accuracy of the HIPP approach with other approaches.

5.1 Experimental Setup

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 [9]. Our solution is called SHIPs (Sat-based Haplotype Inference by Pure Parsimony).

For evaluating SHIPs, as well as the other tools, we have collected a significant number of problem instances. Problem instances of the haplotype inference problem may be obtained following two different approaches:

- **Generate problem instances:** The problem instances are typically generated using Hudson's program `ms` [14]. 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 and a set of haplotypes is generated. Given the haplotypes, genotypes are generated and given to a HIPP solver.

- **Obtain real problem instances:** Haplotypes for small genomic regions have been identified and are available from scientific publications [15, 22, 8]. Additionally, the HapMap project [25] provides a large source of *genotype* data over four populations. Since only genotypes are available, it is not possible to check the accuracy of the HIPP approach on these problems.

[4] See for example [2] for a detailed description of the techniques used for identifying structural properties of genotypes.

Table 2: Results for RTIP, Poly and Hybrid (taken from [3]), 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	15/15	15/15	15/15	15/15
	10	16	50	15/15	12/15	15/15	15/15	15/15
	30	–	50	7/15	11/15	15/15	15/15	15/15
	50	–	30	0/50	27/50	35/50	50/50	50/50
	75	–	30	0/10	4/10	6/10	9/10	10/10
	100	–	30	0/10	3/10	3/10	9/10	10/10
Non-Uniform	10	–	50	15/15	14/15	15/15	15/15	15/15
	30	–	50	15/15	8/15	15/15	15/15	15/15
	50	–	30	8/15	0/15	6/15	14/15	15/15
	75	–	30	2/15	0/15	5/15	6/15	15/15
	100	–	30	0/15	0/15	3/15	4/15	15/15
Hapmap	30:75	–	7:68	0/24	15/24	15/24	17/24	23/24
TOTAL	10:100	0:16	7:68	92/229	124/229	163/229	199/229	228/229

5.2 SHIPs vs RTIP, Poly, Hybrid and Hapar

In order to compare the performance of SHIPs with the performance of other tools based on the HIPP approach, we have used three benchmarks [5]. These benchmarks are 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 considered, a continuous collection of SNPs with a small amount of recombination was selected.

Table 2 provides results for a set of 229 problem instances, including uniform, non-uniform and hapmap instances. The `ms` program has generated 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

[5] These benchmarks were provided by Daniel G. Brown and Ian M. Harrower.

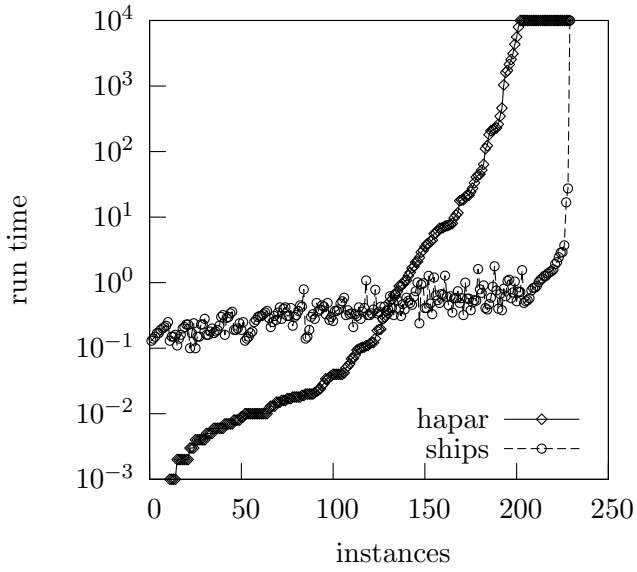


Figure 1: Hapar vs SHIPs on uniform, non-uniform and hapmap instances.

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 2 compares the performance of RTIP [11], Poly [2], Hybrid [4], Hapar [26] and SHIPs on solving these 229 problem instances. For each solver, the number of problems solved within 7200s is given. Results for RTIP, Poly and Hybrid were taken from [3], where a 1.5GHz Intel Pentium 4 with 1GB of RAM running Debian Linux was used. We have applied a conversion factor to the results for Hapar and SHIPs, that compensates for the different speed of the machine used to run SHIPs and the machine used to run RTIP, Poly and Hybrid.

From Table 2 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 30 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, and a different machine was used (1.9 GHz AMD Athlon XP with 1GB of RAM running RedHat Linux). With the only exception of one hapmap problem instance for which both SHIPs and Hapar abort, each of the problem instances was solved by SHIPs in less than 30 seconds. Although Hapar aborted on 20 instances, around 80% of the instances were solved in less than 100 seconds. These results contrast with the results obtained for the other three tools for which, accordingly to [3], many of the instances solved required more than 100 seconds. Moreover, and besides trivial instances, SHIPs is in general faster than Hapar by 1 to 4 orders of magnitude.

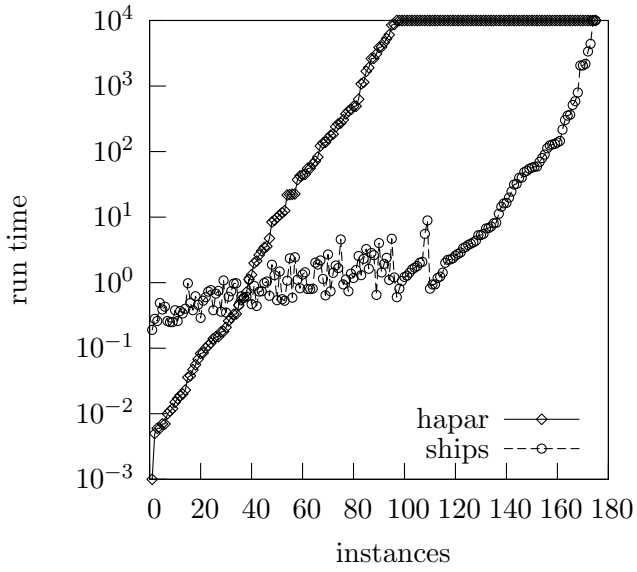


Figure 2: Hapar vs SHIPs on additional uniform and non-uniform instances.

In order to further compare the performance of SHIPs and Hapar, we have performed experiments on additional 175 hard problem instances. Some of the instances were also provided by Brown and Harrower (although results in [3] for RTIP, Poly and Hybrid do not include them): 70 uniform instances with 50 genotypes each (15 instances with 10 sites and recombination level 40, 3×15 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 3×15 uniform and 3×15 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 advantages of the new SAT-based method. 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.

5.3 SHIPs vs Phase and Haplotyper

The results of the previous section are clear. The SHIPs approach is by far the most efficient solution for haplotype inference by pure parsimony. This section compares the accuracy of SHIPs (and so of pure parsimony) with PHASE [24, 23] and Haplotyper [21]. We have selected PHASE and Haplotyper because these tools use approaches for solving the haplotype inference problem that are very different from the pure parsimony approach.

PHASE and Haplotyper are both statistical-based methods following a coalescent approach. A coalescent is a stochastic process that provides a history of how a set of sampled haplotypes in a population has evolved. PHASE uses an infinite-sites model and the Gibbs sampling algorithm, whereas Haplotyper uses a an expectation-maximization and a partition-ligation algorithm. In our experiments, we have used PHASE version 2.1 and Haplotyper version 1.0. (Haplotyper has been configured for 20 rounds, as suggested by the authors.)

We have performed this evaluation using three sets of haplotypes on biological data: B2AR, ACE and CF. These sets of haplotypes are characterized as follows:

- B2AR is a set of 12 haplotypes with 13 SNPs obtained from [8] (see Table 1). These haplotypes correspond to the β_2-adrenergic receptors. The β_2-adrenergic receptors (β_2AR) are G protein-coupled receptors that mediate the actions of catecholamines in multiple tissues.

- ACE is a set of 22 haplotypes with 52 SNPs obtained from [22]. The angiotensin converting enzyme (ACE) is encoded by the gene DCP1 and catalyses the conversion of angiotensin I to the physologically active peptide angiotensin II. Due to this key function in the renin-angiotensin system, many association studies have been performed with DCP1. Rieder et al. [22] completed the genomic sequencing of DCP1 from 11 individuals, and identified 78 varying sites in 22 chromosomes. Fifty two out of the 78 varying sites are non-unique polymorphic sites, and complete data on these 52 biallelic markers are available.

- CF is a set of 29 haplotypes with 23 SNPs obtained from [15]. Cystic fibrosis (CF) is a life-threatening disorder that causes severe lung damage and most people with the disease usually do not live beyond their 40s. Cystic fibrosis is due to a defective gene that causes the secretions to become thick and sticky, and therefore respiratory failure is the most dangerous consequence of this disease.

For each of the three sets of haplotypes, we have generated sets of genotypes with different sizes. Genotypes have been generated by randomly pairing two haplotypes. For each size we have generated 100 problem instances.

For both B2AR and ACE we have generate sets with sizes ranging from 5 to 50 genotypes. We have observed that when using large sets (with size ≥ 30) the error rates are close to zero. For CF we have generated sets with sizes ranging from 4 to 22 genotypes. We should note that to our best knowledge never in the past such an extensive evaluation has been performed. For example, [26] reports results for sets on biological data as large as 11 genotypes, whereas [13] reports results for sets on biological data as large as 13 genotypes.

We have evaluated the accuracy of the three tools according to the error rate. This measure is commonly used in the haplotype inference problem (e.g. see [26, 13]). The error rate gives the proportion of genotypes whose original haplotype pairs are incorrectly inferred. For example, an error rate of 0.15 means that 15% of the genotypes has been incorrectly explained.

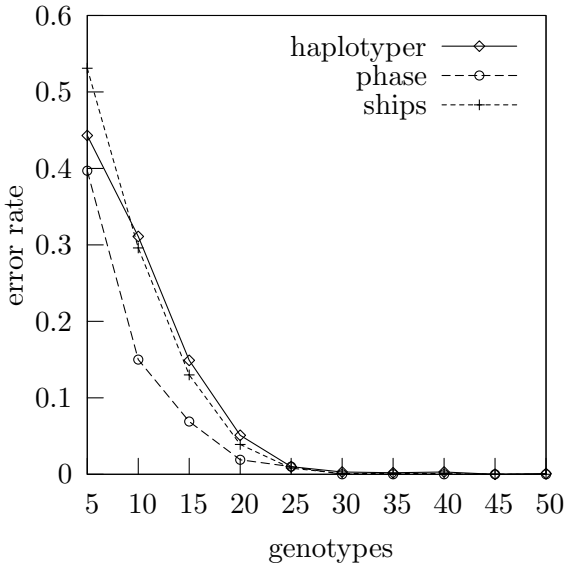


Figure 3: PHASE and Haplotyper vs SHIPs on B2AR instances.

The plots in Figures 3, 4 and 5 give the error rate as a function of the number of genotypes for Haplotypes, PHASE and SHIPs on B2AR, ACE and CF, respectively. Each point is the average error rate for a set of 100 instances having the same number of genotypes. From these figures, it is clear that the error rate decreases as the number of genotypes increases. Observe that as the number of genotypes increases, the probability of having genotypes uniquely explained (with at most one heterozygous site) increases. Most probably, the haplotypes that explain these genotypes also explain other genotypes. This process naturally serves the pure parsimony and the coalescent approach.

The three tools have very similar performances with respect to accuracy. None of them dominates either as the best or as the worst approach. Although PHASE is the most accurate on solving the B2AR instances, it is the less accurate on the ACE instances. With Haplotyper occurs the opposite. For both the B2AR and the CF instances, SHIPs is ranked as second. For the CF instances, the accuracy of SHIPs can be worst than for PHASE and Haplotyper for small size instances but is as good as PHASE for large problem instances. Indeed, SHIPs seems to be the most balanced approach. Given these results, we believe that the HIPP approach is worthwhile investigating further. Since it is most likely that none of the approaches will be clearly better than all the others, results with different tools should be obtained and combined for solving the haplotype inference problem.

These results confirm the results obtained in [26], where the authors claim that the HIPP approach is as accurate as other existing approaches. Nonetheless, in [13] Hapar has been reported as being less accurate than the other tools. However, for these experiments many genotypes could not be resolved by Hapar within two hours and consequently were discarded. Hence, this paper gives, for the first time, results on the accuracy of the HIPP approach on large sets from biological data.

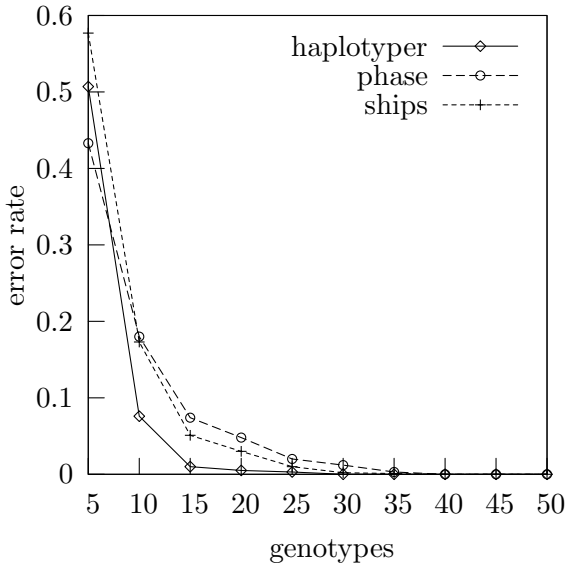


Figure 4: PHASE and Haplotyper vs SHIPs on ACE instances.

6 Conclusions and Future Work

This paper conducts an experimental evaluation on SHIPs, the first SAT-based approach for the problem of inferring haplotypes under the pure parsimony criterion. The SHIPs tool uses a SAT model, incorporates a number of key pruning techniques and uses an iterative algorithm that enumerates the possible solution values for the target optimization problem.

We compared the performance of SHIPs with that of other tools based on the pure parsimony approach. The results presented in the paper are conclusive. SHIPs is much more efficient than all existing combinatorial methods, 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.

We also compared the accuracy of SHIPs with that of other tools based on statistical methods, using a diverse set of biological data. For the first time we have been able to evaluate a tool base on the HIPP approach on a significant number of problem instances on biological data. Experimental results indicate that the HIPP approach is as accurate as other approaches and therefore tools based on the HIPP approach should be considered for performing haplotype inference on biological data.

Although SHIPs is both efficient and accurate on existing problem instances, several challenges still remain. Expected improvements include the development of additional search pruning techniques, more sophisticated lower bounding, and a more optimized encoding into SAT. Moreover, a more extensive experimental evaluation should be conducted, including other tools and more problem instances.

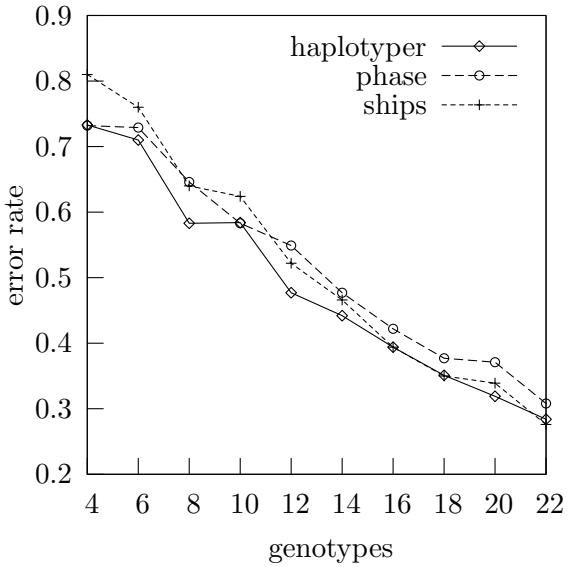


Figure 5: PHASE and Haplotyper vs SHIPs on CF instances.

Finally, the SAT-based solution for the haplotype inference problem allows the utilization of criteria other than pure parsimony or extensions of the pure parsimony criterion. For example, it is simple modify the proposed model to give preference to specific haplotypes (e.g. the haplotypes that can be used to explain the largest number of genotypes).

Acknowledgments

This work is partially supported by Fundação para a Ciência e Tecnologia under research project POSC/EIA/61852/2004. We want to thank the authors of [3, 13] for providing assistance on the results reported on their papers.

References

[1] R. Bayardo Jr. and R. Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the National Conference on Artificial Intelligence*, pages 203–208, July 1997.

[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)*, 2004.

[3] D. Brown and I. Harrower. Integer programming approaches to haplotype inference by pure parsimony, 2005. Submitted.

[4] D. Brown and I. Harrower. A new formulation for haplotype inference by pure parsimony. Technical report, University of Waterloo, School of Computer Science, 2005.

[5] S. Cook. The complexity of theorem proving procedures. In *Proceedings of the Third Annual Symposium on Theory of Computing*, pages 151–158, 1971.

[6] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the Association for Computing Machinery*, 5:394–397, July 1962.

[7] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery*, 7:201–215, July 1960.

[8] 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 Sciencies of the United States of America*, volume 97, pages 10483–10488, September 2000.

[9] N. Eén and N. Sörensson. An extensible sat-solver. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 502–518, 2003.

[10] A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, and T. Walsh. Global constraints for lexicographic orderings. In *International Conference on Principles and Practice of Constraint Programming*, 2002.

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

[12] D. Gusfield and S.H.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.

[13] Y.-T. Huang, K.-M. Chao, and T. Chen. An approximation algorithm for haplotype inference by maximum parsimony. *Journal of Computational Biology*, 12(10):1261–1274, December 2006.

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

[15] B. Kerem, J. Rommens, J. Buchanan, D. Markiewicz, T. Cox, A. Chakravarti, M. Buchwald, and L. C. Tsui. Identification of the cystic fibrosis gene: Genetic analysis. *Science*, 245:1073–1080, 1989.

[16] 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.

[17] I. Lynce and J. Marques-Silva. Efficient haplotype inference with boolean satisfiability. Technical Report 2/2006, INESC-ID, February 2006.

[18] I. Lynce, J. Marques-Silva, and A. Oliveira. Método e sistema para a inferência de haplotipos por parcimónia pura usando satisfação proposicional. Patent PT 103434, February 2006.

[19] J. P. Marques-Silva and K. A. Sakallah. GRASP-A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999.

[20] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Engineering an efficient SAT solver. In *Design Automation Conference*, pages 530–535, June 2001.

[21] T. Niu, Z. Qin, X. Xu, and J. Liu. Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *American Journal of Human Genetics*, 70:157–169, 2002.

[22] M. J. Rieder, S. T. Taylor, A. G. Clark, and D. A. Nickerson. Sequence variation in the human angiotensin converting enzyme. *Nature Genetics*, 22:481–494, 2001.

[23] M. Stephens and P. Scheet. Accounting for decay of linkage desequilibrium in haplotype inference and missing data imputation. *American Journal of Human Genetics*, 76:449–462, 2005.

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

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

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

[27] L. Zhang, C. F. Madigan, M. W. Moskewicz, and S. Malik. Efficient conflict driven learning in boolean satisfiability solver. In *Proceedings of the International Conference on Computer-Aided Design*, pages 279–285, 2001.