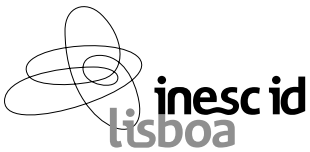
Improved Lower Bounds for SAT-Based Haplotype Inference

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

Technical Report 17/2006

June 2006



Software Algorithms and Tools for constraint solving - SAT

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

Abstract

Mutation in DNA is the principal cause for differences among human beings, and Single Nucleotide Polymorphisms (SNPs) are the most common mutations. Hence, a fundamental task is to complete a map of haplotypes (which identify SNPs) in the human population. Associated with this effort, a key computational problem is the inference of haplotype data from genotype data. Existing solutions for the haplotype inference by pure parsimony (HIPP) problem include Integer Linear Programming (ILP) [6, 1, 2] and branch and bound [13]. Recent work has shown that Boolean Satisfiability (SAT) approaches to the HIPP problem, built on top of modern SAT solvers [3], are remarkably efficient, yielding most often orders of magnitude speedups with respect to previous solutions to the HIPP problem. The objectives of this paper are twofold. The first objective is to provide a brief survey of SHIPs [10, 11], a SAT-based solution to the HIPP problem. An additional objective of the paper is to propose improvements to the SHIPs approach described in [10, 11]. One important aspect of SHIPs is the lower bounding procedure, which reduces the number of iterations of the basic algorithm, but also indirectly simplifies the resulting SAT model. The paper describes a new, much more effective, lower bounding procedure. This new lower bounding procedure is guaranteed to always be as tight as the procedure of [10, 11]. In practice, however, the new lower bounding procedure is in most cases significantly tighter, allowing relevant performance speedups in challenging problem instances.

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 particular focus has been put on the identification of Single Nucleotide Polymorphisms (SNPs), point mutations found with only two possible 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 whether the individual is heterozygotic at that position, i.e. whether the values inherited from both parents are different. 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*.

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) [9]. The Pure-Parsimony problem is to find a solution to the haplotype inference problem that minimizes the total number of distinct haplotypes used. Recent work [12, 10, 11] has proposed the utilization of SAT for solving the HIPP problem. Experimental results are significant: on existing well-known problem instances, the SAT-based HIPP solution (SHIPs) is the *most* efficient approach to the HIPP

problem, being orders of magnitude faster than *any* other alternative exact approach for the HIPP problem.

This paper provides a brief description of the most effective SAT-based model proposed in [12, 10, 11]. In addition, the paper introduces an improved clique-based lower bounding technique. The lower bounding procedure allows reducing the number of iterations of the basic algorithm, and also indirectly simplifying the resulting SAT model.

The paper is organized as follows. Section 2 reviews the problem of haplotype inference, and formalizes haplotype inference by pure parsimony. Afterwards, Section 3 provides a description of the SAT-based approach for the HIPP problem. Section 4 describes the new approach for computing clique-based lower bounding. Section 5 summarizes related work. In addition, Section 6 provides experimental results, comparing the performance of SHIPs with and without the new lower bounding mechanism, and Section 7 concludes the paper.

2 Haplotype Inference

The *genome* is the whole hereditary information of an organism that is encoded in the *DNA*. The DNA is normally packaged in the form of a set of large macromolecules called *chromosomes*. In diploid organisms, chromosomes are grouped in sets of two, where one chromosome is inherited from the father and the other from the mother. In what follows we will only consider diploid organisms.

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. Since each of the bases can only combine with one other base, the bases on the old strand dictate which bases will be on the new strand. Each of the two double strands obtained at the end will end up as a complete replica of the original DNA, unless a mutation occurs.

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 *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.

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. 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 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.

3 SAT-Based Haplotype Inference

This section surveys the SAT-based approach for the HIPP problem originally proposed in [10] and further developed in [11]. The section is organized in three main parts. First the top-level SHIPs algorithm is described. Afterwards, Section 3.2 presents the *core model*, which contains the key ideas of the SAT-based model for the HIPP problem. The core model is ineffective in practice. Hence, a number of key optimizations are detailed in Section 3.3. The resulting (complete) SHIPs model is remarkbly effective in practice.

3.1 The SHIPs Algorithm

The top-level SHIPs algorithm accepts a set of genotypes $\mathcal{G}$ and a lower bound on the number of haplotypes lb necessary to explain the set of genotypes. A trivial value of lb is 1. The algorithm searches for the least value r such that there exists a set $\mathcal{H}$ of haplotypes, with $r = |\mathcal{H}|$, which explain all genotypes in $\mathcal{G}$. Observe that the the value of r is guaranteed to be such that $lb \leq r \leq 2\,n$. A

solution with $2\,n$ haplotypes is guaranteed to exist. For each value of r considered, a CNF (Conjuntive Normal Form) formula φ^r is created, and a SAT solver is invoked (identified by the function call SAT(φ^r)).

The search for minimum number of haplotypes proceeds through increasing values of r, starting with $r = lb$ and terminating when the resulting instance of SAT is satisfiable. The last value of r is returned.

Other organizations of the top-level SHIPs algorithm could be considered. Examples include searching down from an upper bound or performing a binary search between a lower and an upper bound. The motivation for searching up from a lower bound is to ensure that the generated CNF formulas are the *most* compact. With the proposed approach, the largest generated CNF formula is obtained for the number of haplotypes corresponding to the target solution. Alternative approaches would generate larger CNF formulas.

Similarly to [2], genotype and site reduction techniques are applied before generating the CNF formulas.

3.2 The Core 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, $1 \leq k \leq r$. 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, $h_{k\,1} \ldots h_{k\,m}$. A valuation $v : \{h_{k\,1}, \ldots, h_{k\,m}\} \rightarrow \{0, 1\}$ to the bits of h_k is denoted by h_k^v. Observe that valuations can be extended to other sets of variables.

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, $1 \leq i \leq n$. 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^a_{ki} and s^b_{ki}, with $k = 1, \ldots, r$. Hence, genotype g_i is explained by haplotypes h_{k_1} and h_{k_2} if $s^a_{k_1 i} = 1$ and $s^b_{k_2 i} = 1$. Clearly, g_i is also explained by the same haplotypes if $s^a_{k_2 i} = 1$ and $s^b_{k_1 i} = 1$.

If a site g_{ij} of a genotype g_i is either 0 or 1, then this is the value required at this site and so this information is used by the model.

If a site g_{ij} is 0, then the model requires, for $k = 1, \ldots, r$:

$$(\neg h_{kj} \vee \neg s^a_{ki}) \wedge (\neg h_{kj} \vee \neg s^b_{ki}) \tag{1}$$

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 $k = 1, \ldots, r$:

$$(h_{kj} \vee \neg s^a_{ki}) \wedge (h_{kj} \vee \neg s^b_{ki}) \tag{2}$$

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, one requires that the haplotypes explaining the genotype g_i have opposing values at site i. This is done by creating one variable $t_{ij} \in \{0, 1\}$, such that site j of the haplotype selected by the a representative selector assumes the same value as t_{ij}, and site j of the haplotype selected by the b representative selector assumes the complemented value of t_{ij}. As a result, the model requires, for $k = 1, \ldots, r$:

$$\begin{array}{l}(h_{kj} \vee \neg t_{ij} \vee \neg s^a_{ki}) \wedge (\neg h_{kj} \vee t_{ij} \vee \neg s^a_{ki}) \wedge \\ (h_{kj} \vee t_{ij} \vee \neg s^b_{ki}) \wedge (\neg h_{kj} \vee \neg t_{ij} \vee \neg s^b_{ki})\end{array} \tag{3}$$

Observe that h_{kj} equals $\neg t_{ij}$ if $s^a_{ki} = 1$ and h_{kj} equals t_{ij} if $s^b_{ki} = 1$.

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

$$\left(\sum_{k=1}^{r} s^a_{ki} = 1\right) \wedge \left(\sum_{k=1}^{r} s^b_{ki} = 1\right) \tag{4}$$

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 and requiring the output of the adder to be equal to 1. The encoding in CNF of the adder circuit requires the utilization of additional variables v^a_{ki} and v^b_{ki}. The CNF clauses for the adder circuit are straightforward.

Moreover, the space complexity of the model is analyzed in [10] and is asymptotically equivalent to the Poly model of [1]. In practice, the SAT-based model yields significantly more compact representations than the models of [1, 2], since the number of haplotypes considered is bounded by the solution to the HIPP problem.

3.3 The Complete Model

As mentioned before, the core SHIPs model is ineffective in practice. As a result, a number of key improvements have been developed, which are essential for obtaining significant performance gains over existing approaches.

3.3.1 Breaking symmetries on the h variables.

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. Elimination of redundant assignments can be achieved by enforcing an ordering of the Boolean valuations to the haplotypes [1].

[1] See for example [4] for a survey of work on the utilization of lexicographic orderings for symmetry breaking.

Hence, for any valuation v to the problem variables we require:

$$h_1^v < h_2^v < \ldots < h_r^v \tag{5}$$

It is straightforward to enforce each sorting constraint between two haplotypes in linear size on the number of 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 $h_k < h_{k+1}$. The representation of a comparator circuit in CNF is straightforward.

3.3.2 Breaking symmetries on the s variables.

Besides the the symmetries associated with the h variables, it is also possible to eliminate symmetries on the s variables. Observe that the model consists of selecting a candidate haplotype for the a representative and another haplotype for the b representative, such that each genotype is explained by the a and b representatives. Given a set of r candidate haplotypes, let h_{k_1} and h_{k_2}, with $k_1, k_2 \leq r$, be two haplotypes which explain a genotype g_i. This means that g_i can be explained by the assignments $s^a_{k_1 i} s^a_{k_2 i} s^b_{k_1 i} s^b_{k_2 i} = 1001$, but also by the assignments $s^a_{k_1 i} s^a_{k_2 i} s^b_{k_1 i} s^b_{k_2 i} = 0110$.

This symmetry can be eliminated by requiring that only one arrangement of the s variables can be used to explain each genotype g_i. One solution is to require that the haplotype selected by the s^a_{ki} variables always has an index *smaller* than the haplotype selected by the s^b_{ki} variables. This requirement is captured by the following conditions:

$$\left(s^a_{ki} \rightarrow \bigwedge_{k_2=1}^{k-1} \neg s^b_{k_2 i} \right) \;,\; \left(s^b_{ki} \rightarrow \bigwedge_{k_1=k+1}^{r} \neg s^a_{k_1 i} \right) \tag{6}$$

Clearly, each condition above can be represented by a single clause, for each k_1 (or k_2) and i. Moreover, observe that for genotypes without homozygous sites, the upper limit of the first constraint can be set to k and the lower limit of the second condition can be set to k.

3.4 Computing Lower Bounds

As mentioned earlier, trivial lower and upper bounds on the number of haplotypes are 1 and $2\,n$, respectively. This section describes an approach for computing lower bounds on the number of haplotypes. Lower bounds allow reducing the number of iterations of the top-level SHIPs algorithm, but also allow reducing the number of variables and constraints in the model.

The techniques for computing lower bounds rely on information regarding incompatible genotypes. The approach proposed uses a maximal clique for computing a lower bound on the number of required haplotypes. 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 maximize the computed lower bound, the objective is to identify the maximum clique in I. Since this problem is NP-hard [5], we use the size of a maximal clique in the incompatibility graph, computed using a simple greedy heuristic. The genotype with the highest number of incompatible genotypes is first selected. At each step, the genotype selected is one that is still incompatible with all the already selected genotypes, and preference is given to the haplotype with the (statically computed) highest number of incompatible genotypes.

Moreover, we note that the information regarding the lower bound can be used for *reducing* the size of the model, and so it can also potentially reduce the search space. 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.

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.

4 Improving Clique-Based Lower Bounding

The HIPP problem is APX-hard [9]. As a result, the computed lower bounds are not guaranteed to be tight. Hence, for ILP-based approaches for the HIPP problem, the usefulness of non-tight lower bounds is not clear. In contrast, the iterative SHIPs algorithm can gain by using non-trivial lower bounds, even if the lower bounds are not tight. First, the number of iterations of the SHIPs algorithm is reduced. Second, and more importantly, tighter lower bounds allow simplifying the generated instances of SAT (see Section 3).

This section describes a new approach for computing lower bounds for SHIPs. Similar to the procedure outlined in the previous section, a maximal clique is computed. In addition, analysis of the structure of the genotypes allows further increases to the lower bound. The objective of the new procedure is to identify heterozygotic sites which necessarily require at least one additional haplotype given a set of previously chosen genotypes. The procedure starts from the clique-based lower bound (see previous section) and grows the lower bound by searching for these heterozygotic sites among genotypes not yet considered for lower bounding purposes. Since the algorithm starts from the clique-based lower bound, it is guaranteed never to yield a bound less than the clique-based

Algorithm 1 Improving the clique-based LB

```
ImproveLB(G_C)
 1  lb ← |G_C|
 2  Sort genotypes by increasing number of heterozygotic sites
 3  Create set Gset with genotypes in clique G_C
 4  for each g ∈ set of non-clique genotypes
 5        do Let S be the subset of genotypes in Gset compatible with g
 6           if g has heterozygous site and every s ∈ S has the same homozygous site
 7              then
 8                   lb ← lb + 1
 9                   ng ← MergeGenotypes(S ∪ {g})
10                   Gset ← (Gset − S) ∪ {ng}
11  return lb
```

lower bound.

Algorithm 1 summarizes the new lower bound procedure. The procedure MergeGenotypes creates a new genotype from a set of genotypes such that any heterozygous site or a site with genotypes having both 0 and 1 becomes heterozygous. If all genotypes have the same homozygous site, then the merged site keeps the same value. For each genotype g not in the clique, if the genotype has a heterozygous site and all compatible genotypes have the same value at that size (either 0 or 1), then g is guaranteed to require one additional haplotype to be explained. Hence the lower bound can be increased by 1.

The new lower bound procedure runs in polynomial time. A straightforward analysis yields a run time complexity in $\mathcal{O}(n^2\, m)$, by observing that each call to the MergeGenotypes function can involve at most $\mathcal{O}(n)$ genotypes and each pairwise merge runs in time $\mathcal{O}(m)$. Finally, observe that the asymptotic complexity of the new lower bound procedure is the same as the asymptotic run time complexity for generating the SHIPs model. Hence, the practical impact of the new lower bound procedure is expected to be negligible, as illustrated by the experimental results in the next section.

5 Related Work

Over the last few years, a number of authors have proposed optimization models for the HIPP problem, most of which based on ILP [7]. With a few notable exceptions [13], the majority of the proposed models utilize Integer Linear Programming (ILP) [6, 1, 2].

The original ILP model, *RTIP*, has linear space complexity on the number of possible haplotypes [6], 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. Several optimizations have been developed to reduce the size of the problem formulation [6, 7].

In [13], 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 [1], with space complexity in $\Theta(n^2m)$. More recently, Brown and Harrower [2] introduced a new polynomial-size formulation, *Hybrid*, that is a hybrid of the two ILP formulations above and inherits the strengths of both.

A key technique for tackling the HIPP problem consists of using the structural properties of genotypes with the purpose of reducing the search space. Standard techniques include elimination of duplicate genotypes and duplicate and complemented sites [2].

6 Experimental Results

The experimental results provided in [10, 11] demonstrate that the SHIPs approach is far more efficient that existing approaches to the HIPP problem. Table 1 was obtained from [10] and compares the performance of RTIP [6], Poly [1], Hybrid [2], Hapar [13] and SHIPs on solving 229 problem instances, including uniform and non-uniform instances generated using Hudson's program `ms` [8], as well as problem instances obtained from the hapmap project (`www.hapmap.org`) [2]. The table gives the number of problem instances solved within 1000 seconds. Clearly, Hapar and SHIPs are 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.

In what follows we concentrate on the improvements obtained with the new lower bounding procedure. We start by analyzing the improvements to the computed lower bound. Afterwards, we analyze the effect of the new lower bound on the run times of SHIPs. All results shown were obtained on a 1.9 GHz AMD Athlon XP with 1GB of RAM running RedHat Linux.

For this purpose we use problem instances generated using Hudson's program `ms` [8]. This program generates *haplotypes* following a standard coalescent approach. Given the haplotypes, the genotypes are generated by pairing haplotypes either *uniformly* (repeated haplotypes are removed) or *non-uniformly* (repeated haplotypes are not removed and so have a higher probability of being paired).

[2] These instances were 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

The instances are generated as in [2] but are significantly more complex, since all instances considered have 100 genotypes and 100 sites, and are generated non-uniformly and uniformly. For the purpose of this paper, 15 instances were generated uniformly and 15 instances were generated non-uniformly.

6.1 Quality of the Lower Bound

The first experiment consisted in comparing the quality of the lower bounds, either computed with the procedure outlined in Section 3, and referred to as the *clique LB*, or with the new procedure, referred to as the *new LB*. The results are conclusive. The new procedure yields in all cases substantially larger lower bounds. The difference in the value of computed lower bounds can reach a factor close to 2 for a few test cases.

6.2 Effect on the Run Times

Figure 2 shows a scatter plot comparing the run times (in seconds) of SHIPs with the original lower bound procedure and with the new lower bound procedure. Each dot represents a problem instance. A log-scale is used.

As can be concluded, the run times for SHIPs with the new lower bound are in general significantly smaller. The speedup of SHIPs with the new lower bound procedure can reach 2 orders of magnitude. Observe that these results are significant. The version of SHIPs reported in [10, 11] is the only

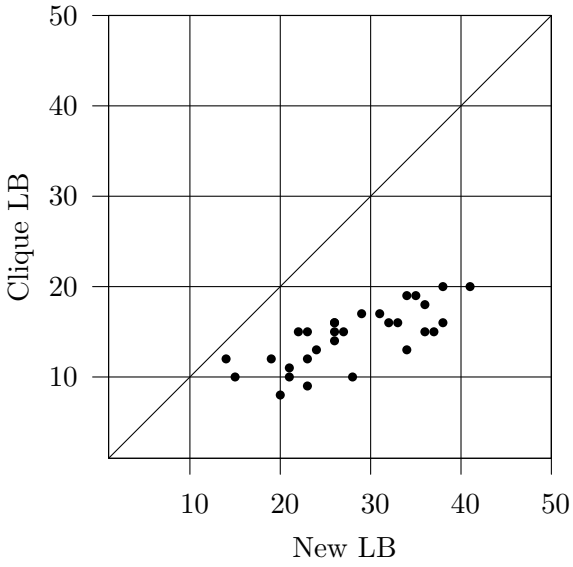


Figure 1: Quality of the computed lower bounds

approach to the HIPP problem capable of solving these problem instances in less than 10000 seconds. Even will all structural simplifications enabled, the ILP models of [6, 1, 2] are unable to solve *any* of these instances in less that 10000 seconds. In addition, Hapar is*only* able to solve 5 of these instances in less than 10000 seconds. As can be observed, SHIPs with the new lower bound solves more than two thirds of these instances in less than 10 seconds, and *only* in one case it requires more than 100 seconds. With three exceptions, the SHIPs with the new lower bound is faster than the previous version. For the three remaining problem instances, the SHIPs with the new lower bound is slower, in part because of the time taken by the new lower bound procedure.

7 Conclusions

This paper surveys SHIPs [10, 11], a Boolean Satisfiability (SAT)-based approach for the problem of haplotype inference by pure parsimony (HIPP), and proposes a new procedure for computing the lower bounds used by SHIPs. Experimental results obtained on challenging problem instances confirm that the proposed lower bound procedure is very effective in practice. Not only are the new lower bounds tighter, but also the new version of SHIPs outperforms significantly the previous version.

The work on SAT-based haplotype inference is fairly recent, and the results already very promising. Further research on SAT-based approaches to the HIPP problem is expected to additional improvements with practical significance are to be expected, as this paper illustrates.

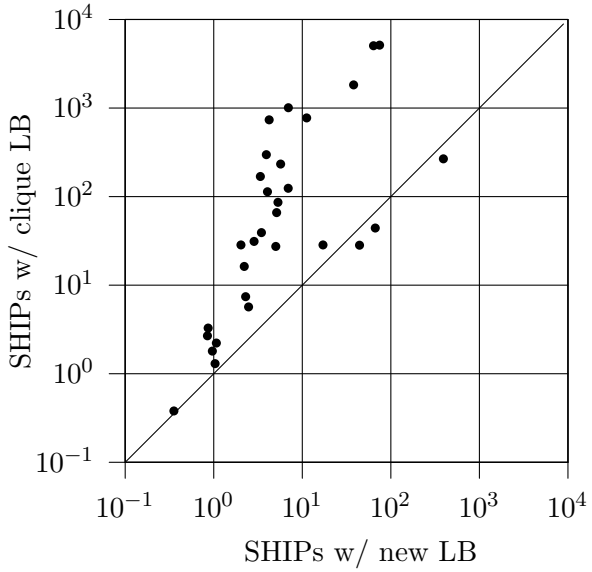


Figure 2: Effect of new lower bound on hard uniform and non-uniform instances

References

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

[2] D. Brown and I. Harrower. Integer programming approaches to haplotype inference by pure parsimony. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(2):141–154, April-June 2006.

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

[4] 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 (CP)*, 2002.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

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

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

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

[9] 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.

[10] I. Lynce and J. Marques-Silva. Efficient haplotype inference with Boolean satisfiability. In *National Conference on Artificial Intelligence (AAAI)*, July 2006. Accepted for publication.

[11] I. Lynce and J. Marques-Silva. SAT in bioinformatics: Making the case with haplotype inference. In *International Conference on Theory and Applications of Satisfiability Testing*, August 2006. Accepted for publication.

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

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