UC Davis Computer Science Technical Report CSE-2003-2

# Haplotype Inference by Pure Parsimony

Dan Gusfield

January 24, 2003

# Haplotype Inference by Pure Parsimony

Dan Gusfield*

January 24, 2003

**Keywords:** haplotype inference, genotypes, SNPs, parsimony, integer programming

**Abstract**

The next high-priority phase of human genomics will involve the development and use of a full *Haplotype Map* of the human genome [7]. A critical, perhaps dominating, problem in all such efforts is the inference of large-scale SNP-haplotypes from raw genotype SNP data. This is called the Haplotype Inference (HI) problem. Abstractly, input to the HI problem is a set of $n$ strings over a ternary alphabet. A solution is a set of at most $2n$ strings over the binary alphabet, so that each input string can be "generated" by some pair of the binary strings in the solution. For greatest biological fidelity, a solution should be consistent with, or evaluated by, properties derived from an appropriate genetic model.

A model that is often mentioned in the literature is the *Pure Parsimony* model, where the goal is to find a *smallest* set of binary strings that can generate the $n$ input strings. The problem of finding such a smallest set is called the *Pure Parsimony Problem*. Unfortunately, the Pure Parsimony problem is NP-hard, and no paper has previously shown how an optimal Pure-parsimony solution can be computed efficiently for problem instances of the size of current biological interest. In this paper, we show how to formulate the Pure-parsimony problem as an integer linear program; we explain how to improve the practicality of the integer programming formulation; and we present the results of extensive experimentation we have done to show the time and memory practicality of the method, and to compare its accuracy against solutions found by the widely used general haplotyping program PHASE. We also formulate and experiment with variations of the Pure-Parsimony criteria, that allow greater practicality. The results are that the Pure Parsimony problem *can* be solved efficiently in practice for a wide range of problem instances of current interest in biology. Both the time needed for a solution, and the accuracy of the solution, depend on the level of recombination in the input strings. The speed of the solution improves with increasing recombination, but the accuracy of the solution decreases with increasing recombination.

## 1 Introduction

Progress on population-scale genomics has focused on the acquisition and use of SNP's and SNP-haplotypes [15, 3]. Consequently, the next high-priority phase of human genomics will involve the development of a full *Haplotype Map* of the human genome [7]. It will be used in large-scale screens of populations to associate specific SNP-haplotypes with specific complex genetic-influenced diseases. Building a Haplotype Map of the human genome has become a central NIH promoted goal [7, 12]. A critical, perhaps dominating, problem in all such efforts is the computational determination of large-scale SNP-haplotypes from raw SNP data.

*Dept. of Computer Science, 3051 Engineering II, University of California, One Shields Avenue, Davis, CA 95616. Email: gusfield@cs.ucdavis.edu Research Supported by NSF grants DBI-9723346 and EIA-0220154

## 1.1 Introduction to SNP's, Genotypes and Haplotypes

In diploid organisms (such as humans) there are two (not completely identical) "copies" of each chromosome (except for the sex chromosome). A description of the data from a single copy is called a *haplotype*, while a description of the conflated (mixed) data on the two copies is called a *genotype*. For many diseases, haplotype data (identifying a set of gene alleles inherited together) is a much better predictor of disease or disease susceptibility than is genotype data.

The underlying data that forms a haplotype is either the full DNA sequence in the region, or more commonly the values of *single nucleotide polymorphisms (SNP's)* in that region. A SNP is a single nucleotide site where exactly two (of four) different nucleotides occur in a large percentage of the population. The SNP-based approach is the dominant one, and worldwide efforts to characterize human molecular genetic variability has resulted in the construction of high density SNP maps (with a density of about one SNP per thousand nucleotides). As this work continues, it is essential to develop methods of deducing haplotypes from SNP data.

We focus on one chromosome with $m$ sites (SNP's) where each site can have one of two states (alleles), denoted by 0 and 1. For each of $n$ individuals, we would ideally like to describe the states of the $m$ sites on each of the two chromosome copies separately, i.e., the haplotype pair for each individual. However, experimentally determining the haplotype pair is technically difficult or expensive. As a result, almost all population data consists only of the set of SNP's possessed by an individual (their genotype), rather than their haplotypes. One then uses computation to deduce haplotype information from the given genotype information. Several methods have been explored and are intensely used for this task [2, 1, 5, 16, 6, 14]. None of these methods are presently fully satisfactory.

Abstractly, for a population of $n$ individuals, input to the haplotyping problem (HI) consists of $n$ *genotype* vectors (or strings), each of length $m$, where each value in the vector is either 0,1, or 2. Each position in a vector is associated with a site of interest on the chromosome. The position in the genotype vector has a value of 0 or 1 if the associated chromosome site has that state on both copies (it is a *homozygous* site), and has a value of 2 otherwise (the chromosome site is *hetrozygous*).

Given an input set of $n$ genotype vectors, a *solution* to the *Haplotype Inference (HI) Problem* is a set of $n$ pairs of binary vectors (strings), one pair for each genotype. For any genotype $g$, the associated binary vectors $v_1, v_2$ must both have value 0 (or 1) at any position where $g$ has value 0 (or 1); but for any position where $g$ has value 2, exactly one of $v_1, v_2$ must have value 0, while the other has value 1. That is, $v_1, v_2$ must be a feasible "resolution" of $g$ into two haplotypes that could explain how $g$ was created. Hence, for an individual with $k$ hetrozygous sites there are $2^{k-1}$ haplotype pairs that can resolve it.

For example, if the observed genotype $g$ is 0212, then the pair of haplotypes 0110, 0011 is one feasible resolutions, out of two feasible resolutions. Of course, we want to find the resolution that actually gave rise to $g$, and a solution for the HI problem for the genotype data of all the $n$ individuals. However, without additional biological insight, one cannot know which of the exponential number of solutions is the "correct one". Therefore, haplotype deduction would be impossible without the implicit or explicit use of some genetic model, either to assess the biological fidelity of any proposed solution, or to guide the algorithm in constructing a solution.

## 1.2 The Pure-Parsimony-criteria

One natural approach to the HI problem that is often mentioned in the literature is called here the *Pure-Parsimony* approach[1]: Find a solution to the HI problem that minimizes the total number of distinct haplotypes used.

[1] This approach was suggested to us Earl Hubbell, who also proved that the problem of finding such solutions is NP-hard [8].

For example, consider the set of genotypes: 02120, 22110, and 20120. There are HI solutions for this example that use six distinct haplotypes, but the solution 00100, 01110; 01110, 10110; 00100, 10110, for the three genotype vectors respectively, uses only the three distinct haplotypes 00100, 01110, and 10110.

The Pure parsimony criteria reflects the fact that in natural populations, the number of observed distinct haplotypes is vastly smaller than the number of combinatorially possible haplotypes, and this is also expected from population genetics theory. Moreover, the parsimony criteria is to some extent involved in existing computational methods for haplotype inference. For example, some papers have tried to explain Clark's method [2] in terms of parsimony [13], although the role of parsimony is not explicit in the method. The haplotyping program PHASE [16] has been partially explained in terms of the parsimony criteria [4]. However, the indirect role of the parsimony criteria in these methods, and the complex details of the computations, makes it hard to see explicitly how the parsimony criteria influences the computation. This makes it difficult to use those methods to evaluate the effectiveness of the parsimony criteria as a genetic model. Moreover, no paper has shown how an optimal Pure-Parsimony solution can be computed efficiently in a practical range of problem instances, although we announced without detail in [6] that integer programming could be used to achieve this result.

In this paper we detail how to compute, via integer-linear-programming, an HI solution that minimizes the number of distinct haplotypes, i.e., is guaranteed to solve the Pure-Parsimony problem. However, the worst-case running time increases exponentially with the problem size, so the empirical issue is whether this approach is practical for problem instances of current interest in population-scale genomics. We improve the practicality of the basic integer programming formulation in a way that is very effective in practice. We report on the results of extensive experimentation we have done to show the time and memory practicality of the method, and to compare its accuracy against existing HI methods that do not explicitly follow the Pure-parsimony criteria. We also formulate and experiment with two variations of the Pure-Parsimony criteria, that allow greater practicality.

Empirically, the end result is that for haplotyping problem instances of current interest, Pure-parsimony *can* be computed efficiently in most cases. However, it's accuracy is somewhat inferior to the solutions produced by the program PHASE, although this depends on the number of sites and the level of recombination.

In more detail, the practicality and accuracy of our approach depend on the level of recombination in the data (the more recombination, the more practical but less accurate is the method). We show here that the Pure-Parsimony approach is practical for genotype data of up to 30 sites and 50 individuals (which is large enough for practical use in many current haplotyping projects). Up to moderate levels of recombination, the haplotype calls are 80 to 95 percent correct, and the solutions are generally found in several seconds to minutes, except for the no-recombination case with 30 sites, where some solutions require a few hours.

While the main point of this paper is to report that Pure-Parsimony solutions can be practically obtained, the accuracy level observed is a validation of the genetic model implicit in the Pure-Parsimony objective function, for a purely randomly picked solution to the HI problem would correctly resolve only a minuscule fraction of the genotypes.

## 2 Pure Parsimony via Integer Linear Programming

We assume that the reader is generally familiar with linear programming: A set of linear inequalities, defined over a set of variables, must be satisfied (by assigning values to the variables), and among all feasible solutions to the inequalities, one seeks a solution that optimizes a linear objective function defined on the same variables (either maximizing or minimizing its value as specified in the formulation). Linear programming problems can be solved efficiently both in theory and practice, and commercial software exists that can efficiently solve linear programs with millions of variables and inequalities. In Integer-linear-programming, one insists that the variables take on values that are

integral. In our application, the values of the variables will be restricted to be either 0 or 1. There are no known methods to solve integer-linear programming problems efficiently in the theoretical worst-case sense, but many integer-programming formulations are efficiently solved in practice, and commercial code exists that efficiently solves huge integer-linear-programs for certain applications. The practical success of integer programming on many problems motivates this empirical study on the use of integer programming to solve the Pure-Parsimony problem.

## 2.1 The conceptual TIP formulation

We will begin by describing a *conceptual* integer-linear-programming solution to the problem of finding an HI solution that minimizes the number of distinct haplotypes used. The solution is conceptual because it would be generally impractical to use. After describing this conceptual solution, we introduce two simple observations that makes this conceptual solution practical on ranges of data of current biological interest.

Let $g_i$ denote the $i$'th genotype input vector, and suppose it has $h_i$ polymorphic sites (i.e., sites with value 2). There are $2^{h_i-1}$ pairs of haplotypes that could have generated $g_i$. In the conceptual integer-linear-programming formulation, we enumerate each one of these pairs, and create one integer-programming variable $y_{i,j}$ for each of the $2^{h_i-1}$ pairs. As we create these $y$ variables, we take note of the haplotypes in the enumerated pairs. Whenever a haplotype is enumerated that has not been seen before, in any of the pairs of haplotypes enumerated for any previously examined genotype, we generate a new integer-programming variable $x_k$ for that haplotype. Thus, no matter how many times that haplotype occurs in the haplotype pairs, over all the genotype vectors, there will only be one $x$ variable generated for it.

Now we explain the linear-programming inequalities defined on the $y$ and the $x$ variables. We will describe these through the following example. For genotype $g_i = 02120$ we enumerate the two haplotype pairs 00100, 01110; and 01100, 00110, and generate the two variables $y_{i,1}$ and $y_{i,2}$ for these pairs. Assuming, that the four haplotypes above (which are all distinct) have not been seen before, we generate the four variables $x_1, x_2, x_3, x_4$ for them. Then, we create the inequality

$$y_{i,1} + y_{i,2} = 1$$

Recall that the variables can only be set to 0 or 1, so this inequality says that in an HI solution, we must select exactly one of the enumerated haplotype pairs as the resolution (explanation) of genotype $g_i$. The $y$ variable set to 1 indicates which haplotype pair will be used in the explanation of genotype $g_i$.

Next, we create two inequalities for *each* variable $y_{i,j}$. These are:

$$y_{i,1} - x_1 \leq 0$$
$$y_{i,1} - x_2 \leq 0$$
$$y_{i,2} - x_3 \leq 0$$
$$y_{i,2} - x_4 \leq 0$$

To explain these inequalities, we examine the first one: $y_{i,1} - x_1 \leq 0$. It says, that if we set $y_{i,1}$ to 1, then we must also set $x_1$ to 1. Essentially, if we select the haplotype pair associated with variable $y_{i,1}$ to explain $g_i$ in an HI solution, then we must use the haplotype associated with variable $x_1$, because that haplotype is one of the pair of haplotypes associated with variable $y_{i,1}$. The next inequality says the same thing for the haplotype associated with variable $x_2$.

These are the two types of inequalities that are included in the integer-programming formulation. We include such inequalities for *each* input genotype vector. Of course, in general, there will be many more inequalities generated than in the above example. If a genotype has $h$ polymorphic sites, then

there will be exactly $2^h + 1$ inequalities generated for it. This fully specifies the inequalities needed in the formulation.

For the objective function, let $X$ denote the set of all the $x$ variables that are generated, over all the genotypes. Recall, that there is one $x$ variable for each distinct haplotype, no matter how many times it occurs in the enumerated pairs. Then the objective function is:

$$\text{Minimize} \sum_{x \in X} x$$

That objective function forces the $x$ variables to be set so as to select the *minimum* possible number of distinct haplotypes. Taken together, the objective function and the inequalities, along with the restriction that the variables can only be set to 0 or 1, specifies an integer-linear-programming formulation whose solution gives an HI solution that minimizes the number of distinct haplotypes used. That is, this formulation truly solves the "Pure-Parsimony" haplotype problem. We refer to this formulation as the TIP formulation. For any specific problem instance, when the TIP formulation for that instance is solved, the set of $x$ variables that are set to 1 in the solution specify a solution to the Pure-Parsimony problem.

# 3 Efficiency

The above formulation can actually be used without further modification for some problems of current biological interest. But for many problems of current interest (up to 50 individuals and 30 sites) the number of inequalities generated would make it impractical to solve the resulting integer program. For that reason, the TIP formulation is only conceptual, and additional ideas are required to make it practical.

The first idea is the following: If the haplotype pair for variable $y_{i,j}$ consists of two haplotypes that are both part of *no* other haplotype pair, then in the integer-program, there is no need to include variable $y_{i,j}$ or the two $x$ variables for the two haplotypes in the pair associated with $y_{i,j}$. Let RTIP denote the integer programming formulation created by removing such $y$ and $x$ variables from TIP. If there is a genotype vector $g$ such that all associated $y$ variables are removed, then there is an optimal solution to the TIP formulation where we arbitrarily choose the haplotype pair for $g$. Otherwise, there is an optimal solution to the original TIP formulation which does not set any of the removed $x$ or $y$ variables to 1. Hence there is no loss in removing them, and the smaller RTIP formulation will find exactly the same optimal solution as the TIP formulation finds.

This idea is particularly effective because DNA sequences in populations have generally undergone some amount of recombination. Recombination is a process where a prefix of one string, and a suffix of another string, are concatenated to form a third string. The haplotypes in a population evolve both by mutations of single nucleotides, and by pairwise recombinations. Depending on the level of recombination that occurred in the evolution the strings, the RTIP formulation can be much smaller (fewer variables and inequalities) than the original TIP formulation. The reason is that as recombination levels rise, the haplotypes become more differentiated, and in turn the genotypes become more different from each other, so that more of the haplotypes enumerated in the TIP formulation only appear in one haplotype pair. These haplotypes are removed in the RTIP formulation. Smaller formulations allow the integer programming solution codes to run more efficiently.

The above idea reduces the size of the integer-programming formulation while preserving the optimal solution. However, if we first enumerate all the haplotypes and haplotype pairs in the TIP formulation, and then remove variables, the work involved could still make the RTIP approach impractical. For a genotype with $h$ polymorphic sites, there are $2^h$ haplotypes, and so brute-force enumeration itself may be very time consuming, even if later many variables are removed. That brute-force enumeration can be eliminated as follows: Let $g_1$ and $g_2$ be two genotype vectors, and

let $H_1$ and $H_2$ be respectively the set of haplotypes that are associated with each genotype, in the conceptual approach. It is easy to identify the haplotypes in $H_1 \cap H_2$, and generate them in time proportional to $m|H_1 \cap H_2|$, where $m$ is the length of the genotype vector. Simply scan $g_1$ and $g_2$ left to right; if a site occurs with a value of 1 in one and 0 in the other, then $H_1 \cap H_2 = \emptyset$; if a site occurs with a 2 in one vector and a 0 or 1 in the other, then set that 2 to be equal to the other value. Then if there are $k$ remaining sites, where both $g_1$ and $g_2$ contain 2's, then there are exactly $2^k$ distinct haplotypes in $H_1 \cap H_2$, and we generate them by setting those $k$ sites to 0 or 1 in all possible ways. The point is that the time for this enumeration is proportional to $m|H_1 \cap H_2|$. Moreover, each generated haplotype in $H_1 \cap H_2$ specifies a haplotype pair that will be included in RTIP, for both $g_1$ and $g_2$.

Any $x$ variable that is included in the RTIP formulation must occur in an intersecting set for some pair of genotypes, and every pair of haplotypes that should be associated with a $y$ variable must also be found while examining some pair of genotypes. Hence we can produce the RTIP formulation, essentially in time proportional to its size, and so if the RTIP formulation is small, we can produce it very quickly.

# 4 A Variation

In the Pure-Parsimony criteria we find a solution to the HI problem that minimizes the *total* number of distinct haplotypes used. However, in most data sets there are individuals whose genotypes are homozygous at every position. These are haplotypes that one knows for sure are in the population, and population genetic theory suggests that they are likely to have been used in pairs of haplotypes that create ambiguous genotypes. In fact, this idea is central in Clark's haplotyping method. So, we should preferentially try to use in the HI solution any haplotypes that appear as homozygous genotypes in the input. That leads to the *Modified-Parsimony* criteria and problem: find an HI solution that minimizes the number of distinct haplotypes used, excluding in the count any haplotype that is seen as a homozygous genotype in the input. Hence this minimizes the number of *new* haplotypes generated, beyond the ones given in the input. Note that the solution to this problem can be smaller than the solution to the Pure-Parsimony problem minus the number of homozygotes in the input. Essentially, we are able to use in the solution any homozygous string from the input, at zero cost. The integer programming formulation for the Modified-Parsimony problem is created by simply removing from the objective function any $x$ variable that is identical to a homozygous genotype in the input.

# 5 Boosting Accuracy

As we will detail below, for experiments with a small number of sites, or a moderate to large level of recombination, the time to generate and solve the RTIP formulation was so small, that we were able to generate multiple optimal solutions, when they existed. We are generally able to find around 100 multiple optimal solutions in under one minute. Prior work and some theoretical considerations, suggested that when multiple optimal solutions are available, a more accurate solution can be obtained by creating the "consensus" solution from the multiple optimals[14]. That is, we simply look at each genotype and use the resolution for it found in the largest number of the multiple optimal solutions found. Consistently, the accuracy of the consensus solution is better than the average accuracy of the multiple optimals.

# 6 Experimental Results

To test the RTIP approach, we ran a widely-used program developed by R. Hudson [9, 10] that uses coalescent theory to generate a simulated population of haplotypes. Haplotypes generated by

the program were then randomly paired to create genotype data. The Hudson program allows one to specify a level of recombination in the simulated data through a parameter $r$. In this paper, we report on the results obtained from data with 50 individuals, with 10 and 30 sites, and with $r$ set to 0, 4, 16, and 40. It is not known what setting of $r$ corresponds to real recombination levels in nature, but the range used here is thought to be large enough to capture realistic levels of recombination.

For each of these eight different settings, we generated 15 data sets and then generated and solved the RTIP formulations (for both the Pure-Parsimony and the Modified-Parsimony criteria). We use the commercial program CPLEX from ILOG to find the optimal integer-programming solutions. We were interested in the size of the generated formulations, and the time needed to solve them, along with the quality of the solutions. We compared the solutions given by the Pure-Parsimony criteria to the solutions given by the Modified-Parsimony criteria (both with the consensus solution when practical), and to solutions obtained by running the publically-available program PHASE, using its default parameter settings.

We summarize the experimental results as follows: First, for ten sites, regardless of the level of recombination, the RTIP formulations were generated and solved in under one second. At low levels of recombination $r = 0$ up to $r = 16$ the quality of the Pure-Parsimony and Modified-Parsimony and PHASE solutions were essentially indistinguishable, with each method occasionally being superior or inferior to the others. The quality of the solutions depended on the level of recombination. For $r = 0$, all methods were correct for 96 to 100 percent of the haplotype calls. For example, both the Pure-Parsimony and PHASE were 100 percent correct in 6 of the 15 trials, and were 96% correct in only one of the 15 trials; they were 98% correct in all the other trials. For $r = 16$, the accuracy ranged from 72% to 96%.

As the recombination level increased, the accuracy of all the methods decreased, but the accuracy is still impressive. For 50 individuals, 10 sites and $r = 40$, the accuracy of all methods ranged from 74% to 96%.

For 30 sites, the practicality of the RTIP approach depends inversely on the level of recombination, while the accuracy of all methods depends directly on the level of recombination. For $r = 0$, the sizes of the 15 datasets were quite variable. One dataset had under 300 variables and the optimal solution was found in 0.03 seconds. Another dataset had 135,000 variables, and the optimal solution was found in 2.5 minutes. The largest dataset had nearly four million variables, and no optimal solution was obtained. A second dataset had two million variables and again no solution was obtained. Most of the datasets had under 10,000 variables and were solved in under two minutes. The accuracy of the 13 datasets that were solved, was in the 80% to 95% range. In comparison, PHASE took much more time, but was slightly more accurate in 10 out 15 datasets (excluding the two where the RTIP formulation wasn't solved), and slightly less accurate in three out of the 15 datasets.

For 30 sites and $r = 16$, the sizes of the RTIP formulations were notably smaller and all the executions ran in seconds. The accuracy was in the 72% to 92% range, but were consistently inferior (but only by small amounts) to the solutions obtained by PHASE.

For 30 sites and $r = 40$, the RTIP formulations again ran in seconds, and the accuracy was in the 50% to 80% range, but PHASE had notably superior accuracy in 13 of the 15 datasets, with accuracy between 68% and 92%.

As expected, the RTIP formulation was generally significantly smaller than the TIP formulation. Two typical instances are: for 50 individuals and 30 sites and $r = 4$, the TIP formulation had 28,580 haplotype variables, while the RTIP formulation had only 5,418; for 50 individuals and 40 sites and $r = 16$, the TIP formulation had 548,352 haplotype variables, while the RTIP formulation had only 129,814.

# 7 A Hybrid Approach

In this paper we have concentrated on datasets with at most 30 sites and 50 individuals. But we have also explored datasets with up to 150 individuals and 100 sites. Datasets of that size are too large for the Pure-Parsimony approach via integer programming. However, we also have an integer programming formulation that builds on Clark's well-known method.

A. Clark [2] suggested a (non-deterministic) algorithm to solve the HI problem. This algorithm is widely used, and is the basis for several large-scale haplotype studies [1, 5, 11]. The algorithm starts with a set of unresolved genotypes and a set of haplotypes that are known to be in the population. The key idea of Clark's algorithm is to repeatedly use the following "resolution rule" until no more unresolved genotypes remain, or no more can be resolved: If an unresolved genotype $g$ can be resolved by a haplotype pair $h, h'$, where $h$ is a haplotype that is already known, or an already deduced haplotype, and $h'$ is its "conjugate" (which is forced, given $h$ and $g$), then declare $g$ resolved with the pair $h, h'$. If $h'$ has not previously been deduced (or known), then add it to the set of available haplotypes that can be used in additional deductions.

In [6] we showed how to maximize, via integer programming, the number of resolutions that Clark's method achieves. By combining that formulation with the ideas in the RTIP formulation, we now can minimize the number of distinct haplotypes used, within the group of solutions that maximizes the number of resolutions. The integer programming formulations produced are much smaller than those produced by RTIP. With this method, we can find solutions in seconds to minutes even for datasets with 100 sites and 150 individuals. For datasets that can be solved by Pure-Parsimony, this hybrid approach is a only a bit less accurate than the Pure-Parsimony solution, so its ability to solve large datasets quickly makes it of interest for further study.

# 8 Conclusions

The parsimony objective function has been frequently mentioned in the haplotyping literature, and is a part of the conceptual underpinning of some existing haplotyping methods. However, because those methods are complex, it is difficult use their performance to access the effectiveness of the Pure-Parsimony objective function. We show here that for genotype data of the size of current interest, the Pure-Parsimony problem can be efficiently solved via integer-linear-programming. However, except in the case when the number of sites is small, these methods, while impressively accurate in an absolute sense, generally produce solutions that are less accurate than those produced by the PHASE program.

While the main point of this paper is to report that Pure-Parsimony solutions *can* be practically obtained, so that the efficacy of the criteria can be accessed, the level of accuracy observed is a validation of the genetic model implicit in the Pure-Parsimony objective function. Purely randomly picked solutions to the HI problem would correctly resolve only a minuscule fraction of the genotypes, in contrast to the observed 80 to 90 percent accuracy of the Pure-parsimony-based method.

# References


[1] A. Clark, K. Weiss, and D. Nickerson et. al. Haplotype structure and population genetic inferences from nucleotide-sequence variation in human lipoprotein lipase. *Am. J. Human Genetics*, 63:595–612, 1998.

[2] Andrew Clark. Inference of haplotypes from PCR-amplified samples of diploid populations. *Mol. Biol. Evol*, 7:111–122, 1990.

[3] M. Daly, J. Rioux, S. Schaffner, T. Hudson, and E. Lander. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29:229–232, 2001.

[4] P. Donnelly. Comments made in a lecture given at the DIMACS conference on Computational Methods for SNPs and Haplotype Inference, November 2002.

[5] M. Fullerton, A. Clark, Charles Sing, and et. al. Apolipoprotein E variation at the sequence haplotype level: implications for the origin and maintenance of a major human polymorphism. *Am. J. of Human Genetics*, pages 881–900, 2000.

[6] D. Gusfield. Inference of haplotypes from samples of diploid populations: complexity and algorithms. *Journal of computational biology*, 8(3), 2001.

[7] L. Helmuth. Genome research: Map of the human genome 3.0. *Science*, 293(5530):583–585, 2001.

[8] E. Hubbel. Personal Communication.

[9] R. Hudson. Gene genealogies and the coalescent process. *Oxford Survey of Evolutionary Biology*, 7:1–44, 1990.

[10] R. Hudson. Generating samples under the Wright-Fisher neutral model of genetic variation. *Bioinformatics*, 18(2):337–338, 2002.

[11] D. Nickerson, S. Taylor, K. Weiss, and A. Clark et. al. DNA sequence diversity in a 9.7-kb region of the human lipoprotein lipase gene. *Nature Genetics*, 19:233–240, 1998.

[12] NIH. Report on variation and the haplotype map: http://www.nhgri.nih.gov/About_NHGRI/Der/variat.htm.

[13] T. Niu, Z. Qin, X. Xu, and J.S. Liu. Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *Am. J. Hum. Genet*, 70:157–169, 2002.

[14] S. Orzack, D. Gusfield, and V. Stanton. The absolute and relative accuracy of haplotype inferral methods and a consensus approach to haplotype inferral. Abstract Nr 115 in Am. Society of Human Genetics, Supplement 2001.

[15] N. Patil and D. R. Cox et al. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, 294:1719–1723, 2001.

[16] M. Stephens, N. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction from population data. *Am. J. Human Genetics*, 68:978–989, 2001.