# Haplotyping as Perfect Phylogeny: Conceptual Framework and Efficient Solutions

[Extended Abstract]

Dan Gusfield
Department of Computer Science
University of California, Davis
gusfield@cs.ucdavis.edu

## ABSTRACT

The next high-priority phase of human genomics will involve the development of a full *Haplotype Map* of the human genome [12]. It will be used in large-scale screens of populations to associate specific haplotypes with specific complex genetic-influenced diseases. A prototype Haplotype Mapping strategy is presently being finalized by an NIH working-group. The biological key to that strategy is the surprising fact that genomic DNA can be partitioned into long blocks where genetic recombination has been rare, leading to strikingly fewer distinct haplotypes in the population than previously expected [12, 6, 21, 7].

In this paper we explore the algorithmic implications of the key (and now realistic) "no-recombination in long blocks" observation, for the problem of inferring haplotypes in populations. We observe that the no-recombination assumption is very powerful. This assumption, along with the standard population-genetic assumption of infinite sites [23, 14] imposes severe combinatorial constraints on the permitted solutions to the haplotype inference problem, leading to an efficient deterministic algorithm to deduce all features of the permitted haplotype solution(s) that can be known with certainty. The technical key is to view haplotype data as disguised information about paths in an unknown tree, and the haplotype deduction problem as a problem of reconstructing the tree from that path information. This formulation allows us to exploit deep theorems and algorithms from graph and matroid theory to efficiently find one permitted solution to the haplotype problem; it gives a simple test to determine if it is the unique solution; if not, we can implicitly represent the set of all permitted solutions so that each can be efficiently created.

## Keywords

haplotype inference, perfect phylogeny, graph realization, graphic matroid recognition

*This is a modestly corrected version of the paper which appeared in RECOMB 2002, pages 166-175* Washington DC, April 18-21, 2002

## 1. INTRODUCTION

Building a Haplotype Map of the human genome has become a central NIH promoted goal [12]. A prototype Haplotype Mapping strategy is presently being finalized by an NIH working-group. The biological key to that strategy is the surprising fact that genomic DNA can be partitioned into long blocks where genetic recombination has been rare, leading to strikingly fewer distinct haplotypes in the population than previously expected [5, 6, 12, 21, 7]. Quoting from [5]:

> In this study we observed strikingly limited haplotype diversity across long distances punctuated by sites of multiple historical recombination events. Essentially, this long genomic region can be parsed into blocks of low diversity in which recombination plays little or no role in assortment of haplotypes... The description of these haplotype patterns suggest powerful methods for testing for association. Specifically, the non-recombining regions with limited haplotype diversity can be treated as single, multi-allelic loci ... (and) are the quantum of genetic variation that should be tested for association.

The lessons for algorithmists are two-fold:

1) Algorithms to determine haplotype structure will be very valuable, because haplotypes will be the "quantum" elements used in disease association studies.

2) In developing algorithms for deducing haplotypes, it is now realistic in important applications (and maybe obligatory in some contexts given the NIH focus on non-recombining regions) to incorporate a no-recombination assumption into the algorithmic methods. Until now, such an assumption would have had to be apologized for, and justified as a way to make the problem tractable.

In this paper we show that the no-recombination assumption is very powerful, and can be exploited to develop efficient methods for deducing all deterministic information about the permitted haplotype solutions.

## 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, and hence of each region of interest. 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*. In complex diseases (those affected by more than a single gene) it is often much more informative to have haplotype data (identifying a set of gene alleles inherited together) than to have only 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 high density SNP maps have been constructed across the human genome with a density of about one SNP per thousand nucleotides.

## 1.2 The biological problem

Because polymorphism screens will be conducted on large populations, it is not feasible to examine the two copies of a chromosome separately, and *genotype* data rather than haplotype data will be obtained, even though it is the haplotype data that will be of greatest use.

Abstractly, data from $m$ sites (SNP's) in $n$ individuals is collected, where each site can have one of two states (alleles), which we denote by 0 and 1. For each individual, we would ideally like to describe the states of the $m$ sites on each of the two chromosome copies separately, i.e., the haplotype. However, experimentally determining the haplotype pair is technically difficult or expensive. Instead, the screen will learn the $2m$ states (the genotype) possessed by the individual, without learning the two desired haplotypes for that individual. One then uses computation to extract haplotype information from the given genotype information. Several methods have been explored and are intensely used for this task [3, 2, 8, 22, 11, 19]. None of these methods are presently fully satisfactory.

## 1.3 The computational problem

Abstractly, input to the haplotyping problem consists of $n$ *genotype* vectors, 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, one pair for each genotype vector. For any genotype vector $g$, 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 "explanation" for the true (but unknown) haplotype pair that gave rise to the observed genotype $g$. Hence, for an individual with $h$ hetrozygous sites there are $2^{h-1}$ haplotype pairs that could appear in a solution to the HI problem.

For example, if the observed genotype $g$ is 0212, then the pair of vectors 0110, 0011 is one feasible explanation, out of two feasible explanations. Of course, we want to find the explanation 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".

# 2. ENTER THE COALESCENT (OR PERFECT PHYLOGENY)

Algorithm-based haplotype inference 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. Most of the models use statistical or probabilistic aspects of population genetics. We will take a more deterministic or combinatorial approach.

The most powerful (and previously, seemingly unrealistic) such genetic model is the population-genetic concept of a *coalescent*, i.e., a rooted tree that describes the evolutionary history of a set of sequences (or haplotypes) in sampled individuals [23, 14]. The key observation is that "In the absence of recombination, each sequence has a single ancestor in the previous generation." [14].

That is, if we follow backwards in time the history of a single haplotype $H$ from a given individual $I$, when there is no recombination, that haplotype $H$ is a copy of one of the haplotypes in one of the parents of individual $I$. It doesn't matter that $I$ had two parents, or that each parent had two haplotypes. The backwards history of a single haplotype in a single individual is a simple path, if there is no recombination. That means that the history of a set of $2n$ individuals, if we look at one haplotype per individual, forms a tree. The histories of two sampled haplotypes (looking backwards in time) from two individuals merge at the most recent common ancestor of those two individuals. (The reason for using $2n$ instead of $n$ will be clarified shortly.)

There is one additional element of the basic coalescent model: the *infinite-sites* assumption. That is, the $m$ sites in the sequence (SNP sites in our case) are so sparse relative to the mutation rate, that in the time frame of interest at most one mutation (change of state) will have occurred at any site. This assumption is almost universally made, and empirical data frequently agrees with this assumption, but does not always agree. How to deal with small deviations from the infinite-sites assumption is an open question.

Hence the coalescent model of haplotype evolution says that without recombination, the true evolutionary history of $2n$ haplotypes, one from each of $2n$ individuals, can be displayed as a tree with $2n$ leaves, where each of the $m$ sites labels exactly one edge of the tree, i.e., at a point in history where a mutation occurred at that site. This is the underlying genetic model that we assume from here on. See [23]

for another explanation of the relationship between sequence evolution and the coalescent model.

For convenience, we assume that we know the ancestral haplotype at the root of the tree, and assign all of its states to be zero[1].

In more computer science terminology, the no-recombination and infinite-sites model says that the $2n$ haplotype (binary) sequences can be explained by a *perfect phylogeny* [9, 10]:

**Definition** Let $B$ be an $2n$ by $m$ 0-1 (binary) matrix. A *perfect phylogeny for* $B$ is a rooted tree $T$ with exactly $2n$ leaves that obeys the following properties:

1) Each of the $2n$ rows labels exactly one leaf of $T$.

2) Each of the $m$ columns labels *exactly one* edge of $T$.

3) Every interior edge (one not touching a leaf) of $T$ is labeled by *at least* one column.

4) For any row $i$, the columns that label the edges along the unique path from the root to leaf $i$ specify the columns of $B$ that have a value of one in row $i$ in $B$. In other words, that path is a compact representation of row $i$.

The classical Theorem of Perfect Phylogeny is that a binary matrix $B$ has a perfect phylogeny if and only if for each pair of columns, there are no three rows with values 0,1; 1,0; and 1,1 in those two columns [9, 10]. Moreover, if the columns of $B$ are distinct, then there is only one perfect phylogeny for $B$. If there are identical columns, then they label the same edge, and we impose no ordering on those labels. Note that an edge to a leaf need not have a column label.

## What happened to the genotype data?

How does the coalescent view of haplotypes relate to the problem of deducing the haplotypes when only the $n$ genotype vectors are given as input?

The answer is that each genotype vector (from a single individual in a sample of $n$ individuals) was obtained from the mating of two of $2n$ haplotype vectors in an (unknown) coalescent (or perfect phylogeny). That is, the coalescent with $2n$ leaves is the history of haplotypes in the *parents* of the $n$ individuals whose genotypes have been collected. Those $2n$ haplotypes are partitioned into pairs, each of which gives rise to one of the $n$ observed genotypes.

[1]This is justified either by standard "out-group" methods of phylogenetics, or by the fact that if the data has even one homozygote, or single site-hetrozygote, we can "declare" that leaf to be the root, and recode all the genotypes relative to that one. The tree built from this may not be historically correct, but it will solve the haplotype problem correctly. If neither of these approaches is possible, we can still find one solution by assigning the ancestral state of any site to be either 0 or 1 depending on which state is more common at that site. This "majority assignment" works because each 2 in a column will ultimately be converted to an equal number of 0's and 1's, and it is well-known that (for a binary matrix) the "majority assignment" of ancestral state works if there is any assignment that works [18].

So, given a set $S$ of $n$ genotype vectors, we want to find a perfect phylogeny $T$, and a pairing of the $2n$ leaves of $T$ which explains $S$. If we are given $T$ and $S$, finding a leaf pairing that explains $S$ is a trivial problem, and hence we need only concentrate on finding $T$.

## 2.1 Further justification for the coalescent model

We have used the "no-recombination in long blocks" observation to justify our no-recombination assumption, leading to the coalescent model. But there are three additional justifications for using this model.

First, one publicly available program, PHASE [22], that deduces haplotype pairs from a set of genotype vectors, incorporates probabalistic consequences of the assumption that the haplotypes evolve according to a coalescent model. Those consequences are used to guide the program, and PHASE is empirically (as claimed in [22] and in my testing) the most successful haplotype inference program to date. The effectiveness of PHASE, compared to programs that don't incorporate any coalescent assumptions is an indirect validation of the underlying model. It says that when you incorporate the coalescent assumption, you are better able to generate solutions that conform to real world data, suggesting that real world data itself fits the coalescent model. Second, the coalescent model of haplotype evolution is assumed in other work in statistical genetics [17]. However, these two approaches only use statistical consequences that are derived from the the coalescent model. Here, we find what can be deterministicly deduced about underlying coalescent.

Finally, often in the biological literature, where people report the haplotypes that they have found (by whatever means), the found haplotypes are displayed in a tree that demonstrates the evolutionary (or mutational) relationship of the haplotypes. Those trees are often perfect phylogenies for the set of haplotypes. An example is shown in [15]. Note that the approach in that literature is to first determine the haplotypes by some means, and then to lay out the tree. The approach in this paper is the reverse: we try to determine the haplotypes from genotype data by seeing how the genotype data deterministicly forces features of a tree, and use those features to deduce haplotype information.

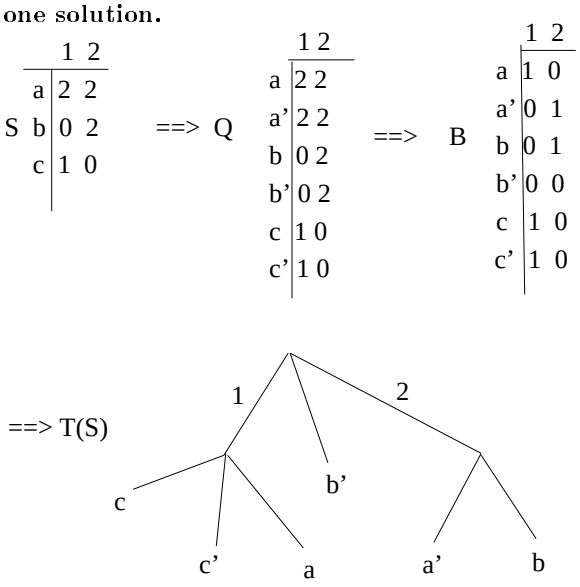
## 2.2 Finally, the purely combinatorial view

Under the coalescent model of haplotype evolution, the general HI problem is now much more constrained, so that only some solutions to the HI problem are permitted. The haplotype inference problem now has precisely the following combinatorial interpretation:

**The Perfect Phylogeny Haplotype (PPH) Problem**

Given a (genotype) matrix $S$ of $n$ rows with values 0, 1 or 2, duplicate each row to create pairs of rows $i, i'$ for $i$ from 1 to $n$. The resulting matrix is denoted by $Q$. Then for each such pair $i, i'$, and every column c where $Q(i, c) = Q(i', c) = 2$, we are required to set *exactly* one of those two cells to have value 0 and the other to have value 1, so that the resulting binary matrix $B$ has a Perfect Phylogeny $T(S)$. Such a setting of values, and the associated perfect phylogeny $T(S)$, is a solution to the PPH Problem. The binary vectors in the

**Figure 1: A simple example where the HI problem has two solutions, but the PPH problem has only one solution.**




two rows $i, i'$ of the solution are then the haplotype pair for individual $i$. By the Perfect Phylogeny Theorem, the setting of values must avoid creating two columns with three rows containing 0,1; 1,0; and 1,1.

In addition to efficiently finding one solution to the PPH problem, we would like to determine if that is the unique solution, and if not, we want to efficiently represent the set of all solutions, so that each one can be generated efficiently.

See Figure 1 for a trivial example. In that example, there are two solutions to the HI problem, but only the one shown solves the PPH problem, i.e., is a solution that has a perfect phylogeny. So in this example, we know for sure what the underlying haplotypes are (assuming the coalescent model of haplotype evolution).

We assume from here on, that for any $S$, there is at least one solution to the PPH problem for $S$, and denote a solution by $T(S)$.

The PPH Problem is similar to the problem considered in [20] where a ternary matrix is also given. In that problem, one must change each 2 to either a 0 or a 1, so that the resulting binary matrix has a perfect phylogeny. An elegant polynomial-time algorithm is presented in [20] for that problem. However that problem is not identical to the PPH problem because of the required row duplications and the associated constraints in the PPH problem.

# 3. SOLVING THE PPH PROBLEM

At first, it may seem that the mating process that forms $n$ genotype vectors from $2n$ haplotype vectors will so disguise the underlying tree structure, that it would be impossible to deterministicly learn much about it. However, the genotype data encodes certain path information from the underlying tree, allowing efficient deterministic deduction of tree features, often the complete underlying tree. In this section we show how this can be done.

## 3.1 Simple tools

The values in $S$ encode certain deterministic information about paths in the underlying perfect phylogeny $T(S)$ from which the genotype data $S$ was created. We recover path information from $S$ with three simple observations:

1. For any row $i$ in $G$, the set of 1 entries in row $i$ specify (without order) the exact set of edge labels on the path from the root to the least common ancestor of leaves $i$ and $i'$, in *every* perfect phylogeny for $S$.

2. For any row $i$ in $S$, and any column $c$, if $S(i, c)$ is 2, then the edge labeled with $c$ must be on the path from the root to *exactly one* of the leaves $i$ and $i'$ in *every* perfect phylogeny for $S$. Another way to say that, is that the path between $i$ and $i'$ must contain the edge with label $c$.

3. For any row $i$ in $S$, and any column $c$, if $S(i, c)$ is 0, then the edge labeled with $c$ must *not* be on the path from the root to either leaves $i$ or $i'$, or on the path between them, in *any* perfect phylogeny for $S$.

The first observation allows us to deduce the labels of the edges on the path from the root to the least common ancestor of $i$ and $i'$, but without their order. However, the order must be the same in every perfect phylogeny, and can be simply deduced as we show next.

Define a *leaf-count* for each column c as follows: Each 1 in column $c$ contributes a count of 2, and each 2 in column $c$ contributes a count of 1. Let $t(c)$ be the total leaf-count for $c$. From the three observations above, we see that in *every* perfect phylogeny $T(S)$ for $S$, the edge labeled with $c$ must have exactly $t(c)$ leaves in its subtree. Therefore, all labels that appear together on the same edge of a perfect phylogeny $T(S)$, must have the same leaf-count. In fact, all edge labels that appear together on the same edge must have exactly the same entries in their respective columns of $S$ (this is a necessary but not sufficient condition for being on the same edge). Similarly, the leaf-counts of labels along successive edges on any directed path from the root must strictly decrease. Hence, if we know which labels appear on a directed path, we also know how they should be organized into edges, and how the edges should be ordered.

We next observe that the paths defined by the 1 entries form an "initial" unique perfect phylogeny that appears in every perfect phylogeny for $S$. Let $C_1$ be the set of columns that each contain at least one 1 entry.

**Lemma** In any perfect phylogeny $T(S)$ for $S$, no path from the root can encounter an edge labeled with a column in $C_1$ if it has already encountered an edge labeled with a column not in $C_1$.

**Proof** Suppose to the contrary that there is in $T(S)$ a path from the root containing a label $e$ for a column not in $C_1$ followed by a label $f$ for a column in $C_1$. Since $f$ is in $C_1$,

there must be a row $i$ with value 1 in column $f$, and hence both leaves $i$ and $i'$ must be below the edge labeled $f$ in $T(S)$. But then both leaves $i$ and $i'$ are below the edge labeled $e$, meaning that row $i$ should have a value of 1 in column $e$, contradicting the assumption that column $e$ is not in $C_1$.

As a consequence of the Lemma, the edges labeled by the columns in $C_1$ appear in a unique "initial" (or upper) perfect phylogeny in every $T(S)$ (assuming no ordering of the labels that appear together on the same edge), and the root of that subtree must be the root of every perfect phylogeny $T(S)$. Further, if $i$ is a row in $S$ that contains no 2 entry, then we know that the complete paths to both leaves $i$ and $i'$ must be in this initial perfect phylogeny.

Let $R_1$ be the set of rows where each contains at least one 1 entry, and let $R_{1,1}$ be the subset of $R_1$ consisting of those rows in $R_1$ that have no 2 entries. We find the "initial" perfect phylogeny for $S$ by first finding, for each row $i$ in $R_1 - R_{1,1}$, the ordered path to the least common ancestor of leaves $i$ and $i'$ as explained above. For each row $i$ in $R_{1,1}$, then we create two ordered paths, one to leaf $i$ and one to leaf $i$. The initial perfect phylogeny is made up of these ordered paths. We can then simply merge the identical initial segments of all these paths to create the unique initial perfect phylogeny $T_1$, containing all the edges labeled by columns in $C_1$, and all the leaf-pairs $i, i'$ for every row $i$ in $R_{1,1}$.

As a simple consequence we have

**Lemma** If each column of $S$ has at least one 1, then there is a unique perfect phylogeny $T(S)$ for $S$, and it can be found efficiently. That is, the underlying $2n$ haplotypes and the mating that gave rise to the $n$ genotypes in $S$, can be completely and deterministicly deduced.

**Proof** The initial perfect phylogeny $T$ for $S$ is unique and contains an edge labeled by $c$ for every column $c$ in $S = C_1$. Moreover, $T$ will contain the leaf-pair $i, i'$ for every row $i$ in $R_{1,1}$. For any row $i$ not in $R_{1,1}$ there must be a contiguous path in the tree containing the column labels for the 2 entries in row $i$, otherwise there is no perfect phylogeny for $S$. So the addition of the leaf pair $i, i'$ onto $T$ is forced, and the resulting tree is $T(S)$ for $S$.

Given the Lemma, the main issue then is how to deal with the mutations corresponding to columns not in $C_1$, and how those mutations extend $T_1$.

## 3.2 Complex tool: Graph Realization

The three path observations stated above will allow us to solve the PPH problem by reducing it to the classical problem of "recognizing graphic binary matroids" [24]. However, we will discuss that problem directly in terms of paths and trees as is done in [1]. This approach does not describe the classical work in its proper historical or mathematical context, but it allows a much shorter exposition.

Let $E_r$ a set of $r$ distinct integers. A "path-set" is an *unordered* subset $P$ of $E_r$. A path-set is "realized" in a undirected, edge-labeled tree $T$ consisting of $r$ edges, if each edge of $T$ is labeled by a distinct integer from $E_r$, and there is a contiguous path in $T$ whose labels consist only of the integers in $P$. Note that since $P$ is unordered, it's presentation does not specify or constrain the order that those edges appear in $T$. In quite different terms, from the 1930's to the 1960's Whitney and Tutte and others studied and solved the following problems:

**The Graph Realization Problem** Given $E_r$ and a family $\Pi = P_1, P_2, ..., P_k$ of path-sets, find a tree $T$ in which each path-set is realized, or determine that no such tree exists. Further, determine if there is only one such $T$, and if there is more than one, characterize the relationship between the realizing trees.

There are elegant mathematical results for each of these problems.

Building on the mathematical results, algorithms for the graph realization problem were studied by several people, (Tutte, Lofgren, Cunningham, Edmonds, Bixby, Wagner, Gavril, Tamari, Fujishige) in the 1950's and work continued until the 1980's. Space limits a full discussion of the history. One of the first algorithms, by Lofgren, finds a realizing tree $T$ by an algorithm whose obvious implementation runs in exponential time. Later, Bixby and Wagner [1] showed how it could be implemented in almost linear time. Hence the problem of finding one realizing tree $T$ for $\Pi$ can be solved efficiently.

## 3.3 Reducing PPH to the graph realization problem

In this section we show, given an instance $S$ of the PPH problem, how to efficiently create a instance $\Pi$ of the graph realization problem such that any realizing tree for $\Pi$ can be used to obtain a solution to the PPH problem for $S$. This then establishes that the PPH problem can be solved efficiently; that one can efficiently test if that solution is unique; that the set of all solutions can be succinctly represented in a compact data-structure; and that each solution can be created in linear time per solution.

Given a genotype matrix $S$, the integers $E_r$ will consists of every column label from $S$, plus several new labels for "glue edges". After a realizing tree $T$ is constructed, the glue edges will be contracted to give a perfect phylogeny for $S$.

## 3.4 Realizing the initial perfect phylogeny

Assume $C_1$ is non-empty. As observed earlier, the edge labels of columns in $C_1$ form the initial perfect phylogeny $T_1$ that is in every solution to the PPH problem for $S$. Tree $T_1$ also contains the leaves $i$ and $i'$ for each row $i$ that does not contain a 2 entry. We could contract $T_1$ and create a new PPH problem, but instead we will reduce the entire PPH problem to a single graph realization problem. So we first show how to create path-sets for $\Pi$ so that any realizing tree for $\Pi$ constructs $T_1$.

For a fixed leaf $v$ of $T_1$, let $e_1, e_2, ...e_x$ be the edge labels in order on the path from the root of $T_1$ to leaf $v$. We extend

that ordered path by adding a "common" glue edge $g_0$ to the start of it. If $v$ is not a leaf $i$ or $i'$ corresponding to a row $i$ in $R_{1,1}$, then we also add a "specific" glue edge $g_v$ to the end of the ordered path. Then, we add to $\Pi$ the path-set $\{g_0, e_1\}$ and the path-set $\{e_i, e_{i+1}\}$ for each $i$ from 1 to $x-1$. Next, add to $\Pi$ the path-set $\{g_0, e_1, e_2\}$, and the path-set $\{e_i, e_{i+1}, e_{i+2}\}$, for each $i$ from 1 to $x-2$. If $g_v$ exists, then we also add the path sets $\{e_x, g_v\}$ and $\{e_{x-1}, e_x, g_v\}$ to $\Pi$.

In the case that the path to $v$ in $T_1$ only contains a single edge, then only put into $\Pi$ the path set $\{g_0, e_1\}$, along with $\{e_1, g_v\}$ and $\{g_0, e_1, g_v\}$ if $g_v$ exists.

It is easy to prove inductively that the only tree realization for these path-sets must be the ordered path $g_0, e_1, ..., e_x$, followed by $g_v$ if it exits.

By creating similar path-sets for each leaf $v$ in $T_1$ (each time with the same $g_0$ but with a different $g_v$), we create a $\Pi$ whose only realizing tree is $T_1$, with the added edge $g_0$ attached to its root, and added edges $g_v$ attached to certain leaves $v$. Let $T^*$ denote this tree.

The purpose of the common glue edge $g_0$ is to glue together all the directed paths at the root of $T_1$, and the purpose of each specific glue edge $g_v$ is to allow future paths not in $T_1$ to be glued to leaf $v$ if needed. We may not actually need them all, but they do no harm at this point.

## 3.5 The rest of the reduction

Now consider a row $i$ of $S$ that contains some 2 value(s). If at least one of those 2 values is in a column $j$ of $C_1$, then in every solution to the PPH problem for $S$, the path between leaves $i$ and $i'$ must go through the edge labeled $j$, so we only need to add to $\Pi$ a path-set consisting of all the column labels in $S$ where row $i$ has a value of 2. The realizing path for the 2 entries of row $i$ will contain an edge in $T^*$ and hence will be glued to $T^*$ without the use of glue edges.

However, if row $i$ has no 2 value in a column of $C_1$, then in any solution to the PPH problem, the path between $i$ and $i'$ cannot go through an edge labeled with a column of $C_1$, and hence must go through a leaf of $T_1$. That leaf is easy to identify, as it is the only leaf $v$ in $T_1$ whose entering edge has a label where row $i$ has a value of 1 in $S$. So, for such a row $i$, we add to $\Pi$ the path-set consisting of all the column labels in $S$ where row $i$ has a value of 2, along with the specific glue edge $g_v$. The use of $g_v$ will ensure that the path realized is attached at the proper location in $T^*$.

As a special case, if $C_1$ is empty, then we create one glue edge $g_0$ and add it to the path-set for each row $i$, which contains the column labels for the 2 entries in row $i$.

After a realizing tree $T$ has been created from $\Pi$ we add in the needed leaf labels as follows: For any row $i$ in $R_{1,1}$, the two leaves $i$ and $i'$ were already in $T_1$ (and hence $T^*$ and $T$), and there is nothing more to do. For a row $i$ that does contain some 2 values, we find the endpoints in $T$ of the path realizing the path-set created for $i$, and add one leaf edge leading to $i$ at one end, and one leaf edge leading to $i'$ at the other edge. Finally, we contract the glue edges to obtain a solution $T(S)$ to the PPH problem for $S$. From this solution, the edge labels on the path from the root to leaf $i$ specifies one of the haplotypes for genotype $i$, while the edge labels on the path to $i'$ specifies the other haplotype.

## 3.6 Uniqueness of the solution

The uniqueness question for the general graph realization problem was resolved (but in quite different terms) by Whitney [26] in 1932. Given a tree $T$ in which all the path-sets of input set $\Pi$ are realized, add a new edge between the two endpoints of path $P_i$ in $T$, for each $P_i$ in $\Pi$. Call these new edges "scaffold edges". Contract any edge in $T$ that is not labeled, and call the new graph $G(T)$. Then $T$ is the only labeled tree that realizes the path-sets in $\Pi$ if and only if $G(T)$ is three-connected. A connected graph is three-connected if there are no two nodes whose removal disconnects the graph.

Now for the PPH problem with input $S$, we had to specify not only path-sets involving columns from $S$, but also involving new glue edges, in order to specify where those paths went. Otherwise, the tree would not be a solution to the PPH problem. So to use Whitney's result to check for uniqueness of a PPH solution, we don't use the tree $T(S)$ but rather use the tree that realizes the complete family of path-sets $\Pi$ as specified in the reduction discussed earlier. Also, it may be possible that the removal of the two endpoints of edge $g_0$ disconnects the graph. In Whitney's sense, this shows that there is another solution (and this will be clearer after the next section), but since we contract glue edges to finally obtain our PPH solution, that "new" solution will not be different. Hence when looking for a pair of nodes whose removal might disconnect the graph, we do not consider the pair consisting of the endpoints of $g_0$.

This gives the following simple test for uniqueness to a solution of the PPH problem (however obtained). Build the perfect phylogeny $T(S)$ associated with the haplotypes in the solution, extend it with the glue edges that would be specified by the reduction, add in all the scaffold edges from the family of path-sets $\Pi$ specified by the reduction; contract any unlabeled edge in $T(S)$. Call this graph $H(T, S)$ and test if there is a pair of nodes (excluding the pair consisting of the endpoints of $g_0$) in $H(T, S)$ whose removal disconnects the graph. If so, then the solution is not unique, otherwise it is. There are conceptually easier specializations of this method that will be detailed in the full version of this paper. Note, graph $H(T, S)$ is two-connected, i.e., that no single node removal will disconnect it.

## 3.7 Multiple solutions

When the solution to the PPH problem is not unique, we would like to understand the relationship between the various solutions, to efficiently represent them all, and to enumerate each one in linear time per solution.

Given a two-connected graph, we define a *two-partition* as a partition of the *edges* of $G$ into two connected subgraphs $G_1$ and $G_2$ of $G$, which each have at least two edges, and which have exactly two nodes (denoted $x$ and $y$) in common. Note that both $G_1$ and $G_2$ contain a copy of both $x$ and $y$. $G$ is not three-connected because the removal of $x$ and $y$ would disconnect $G$. If we connect node $x$ in $G_1$ to node $y$ in $G_2$,

and node $y$ in $G_1$ to node $x$ in $G_2$, this results in a new edge-labeled graph $G'$. This operation is called a "twisting" around the two-partition. It may happen that $G$ and $G'$ are isomorphic as unlabeled graphs, but as labeled graphs they are different. However, a set of edge-labels is contained in a cycle in $G$ if and only if that set is contained in a cycle in $G'$. Therefore, if $H(T, S)$ is defined as above, and $G'$ is obtained from $H(T, S)$ by a twisting around a two-partition, then $G'$ is the graph $H(T', S)$ for a tree $T'$ (containing the same edge labels as $T$) that realizes all the path-sets realized by $T$. Therefore, a new solution to the PPH problem for a given $S$ can be obtained by this twisting operation around the two-partition. Again, we are not interested in twisting around the endpoints of edge $g_0$, because when we later contract glue edges, the resulting solution is the same.

In 1933 Whitney showed that the converse of this twisting result is also true for two-connected graphs [27]. This theorem can be adapted[2] to show that if $T$ and $T'$ are different edge-labeled trees that realize the same path-sets, then $H(T', S)$ can be obtained from $H(T, S)$ by a series of twisting operations. So all the solutions to the PPH problem for a given $S$ are related in this way only.

Further, the set of all such solutions can be implicitly represented in a data-structure that can be built in linear time (once a first solution is found). The data-structure we use is related to one developed by by Cunningham and Edmonds [4, 1] to represent all realizing (non-isomorphic) trees. Without explicitly discussing graph realization, Hopcroft and Tarjan [13] also developed a related data structure, and a linear time algorithm to produce it.

Starting with graph $G = H(T, S)$, and assuming it is not three-connected, we find a two-partition of $G$ into subgraphs $G_1$ and $G_2$ as defined previously. Denote the separating nodes as $x$ and $y$. In both $G_1$ and $G_2$ we add a new edge between $x$ and $y$, give it the same new label, and call it a "marker edge". The two resulting graphs are called "split graphs". We replace $G$ by those two split graphs, and recurse. That is, if a two-partition can be found in a component of the current graph, we use it to create two new split graphs, and replace the component by those two split graphs. This is continued until no further splitting is possible. An example of this is shown in Figure 3. This is essentially the splitting procedure described in [13]. However, in that paper, certain components are merged after all the splitting is done. In our application we also merge certain components, but use a different merging rule, for a different end result. Merging is the inverse of splitting - if two components have the same-labeled marker edge, then we attach the components at the endpoints of the marker edge (without twisting), and remove the marker edge. In our application, if we have a component that contains no labeled tree edges (labeled with a column of $S$), then we merge it together with another component, and continue this until all components have at least one edge labeled with a tree-edge.

The finished data-structure represents all the solutions to the PPH problem for $S$. To enumerate the solutions, independently decide for each marker edge, whether to twist one component containing it around that marker edge, or not. That is, decide whether to re-attach the end-points of the marker edge as in their original orientation, or to reverse the attachment. For example, when no twists are made, we obtain the original tree embedded in $H(T, S)$.

[2] All the expositions I have seen only concern graphs that are non-isomorphic when unlabeled, but it is easy to verify that the theorem holds also for differently labeled, but isomorphic graphs, which is what we need for the PPH problem.

**Figure 2: An example genotype matrix $S$ we will use to illustrate the enumeration of multiple solutions.**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| a | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| b | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| c | 2 | 2 | 0 | 0 | 2 | 0 | 2 |
| d | 2 | 2 | 0 | 0 | 0 | 2 | 0 |

If there are $t$ marker edge-labels in the data-structure, then there are exactly $2^t$ distinct solutions to the PPH problem.

Figure 2 gives an example genotype matrix $S$. The splittings used to create the data-structure for $S$ are shown in Figure 3, and the final data-structure is shown in Figure 4. One of the eight permitted solutions, after twisting around markers edges B, C, and D, is shown in Figure 5.

## 4. FUTURE WORK

This paper is intended to show that the PPH problem has an efficient solution, via reduction to the graph realization problem. However, the efficient known algorithms for graph realization are complex and difficult to implement. It is clear that many simplifications can be made for the special case of the PPH problem, and future work will find such simplifications before any programming is done.

One of the purposes of having an algorithm that can deterministicly deduce haplotype information under the coalescent model, is to explore the conditions under which the PPH solution is unique. Those conditions will guide the design of haplotype screens, for we want to be able to uniquely determine the haplotypes after the genotypes are collected. As one condition, it seems reasonable that as the ratio of individuals to sites increases, the number of permitted solutions decreases, and we conjecture that for randomly generated data under the coalescent model, when the number of individuals is above $m \log m$ (where $m$ is the number of sites), the probability that the solution is unique will be high. The hand-simulations we have done (using input generated by a widely-used simulation program that produces haplotypes under the coalescent model [14]), are consistent with this conjecture. Once a fully working program is finished (and this is not simple) we will be able to study this issue in depth. It is important because the answer to that question will guide the design of haplotyping studies: how many people do you need when you have a given number of sites of interest, and conversely, if you have a fixed number of people, how many sites can reliably be handled.

We also conjecture that the use of the data-structure to represent all the solutions to the PPH problem will lead to a

Figure 3: The splitting steps used to build the data structure to represent all the solutions to the PPH problem given in Figure 2. A solution to the PPH problem is shown in 1), along with the dashed scaffold edges. There are four splitting steps in building the data-structure, shown in 2), 3), 4), 5). The $T_1$ portion of the data-structure is simplified in 2), and omitted in the remainder of the figure. The marker edges are displayed as thick dashed edges, and are labeled with capital letters. To create the final data-structure, the two components with the A marker edge are merged. The finished data-structure is shown in the next figure.

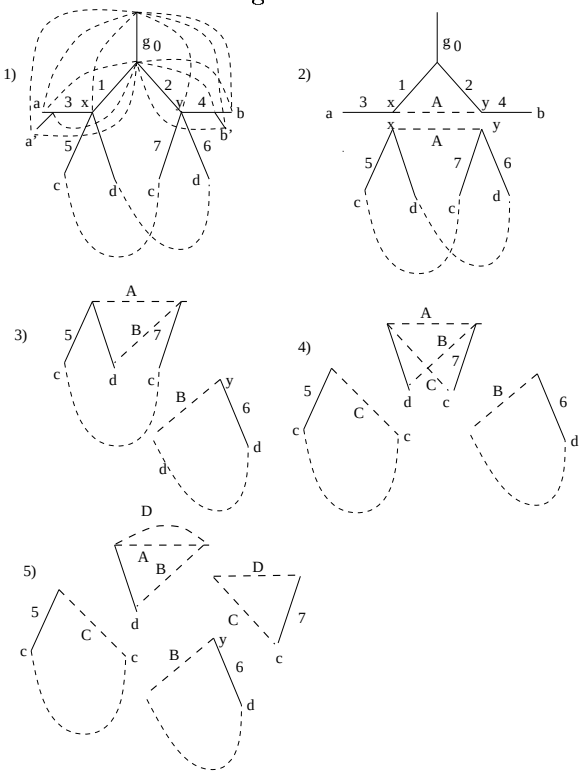



Figure 4: The finished data-structure representing all the solutions to the PPH problem. To enumerate the solutions, we keep one component fixed (the top component say), and then independently decide how to orient the three marker edges in the other components. Hence there are eight solutions to this PPH problem. When we twist and reconnect the graph, the resulting tree may have internal edges without labels, or an internal node with only one out tree edge. In either case, we contract those edges to form a single edge. Also, it is difficult to keep track of the leaf labels during twisting and reconstruction. The easiest approach is to drop, for each row $i$, any distinction between leaf label $i$ and $i'$, and label them both $i$. The two $i$ leaves are then located at the ends of the scaffold edge that ran between $i$ and $i'$; wherever that scaffold edge ends up after the twisting, defines the two $i$ leaves. In a sense, we first label the scaffold edge with $i$ during the twistings. If either end is located at an internal node after the twistings, just create a new leaf edge with no edge label to a new leaf labeled $i$.

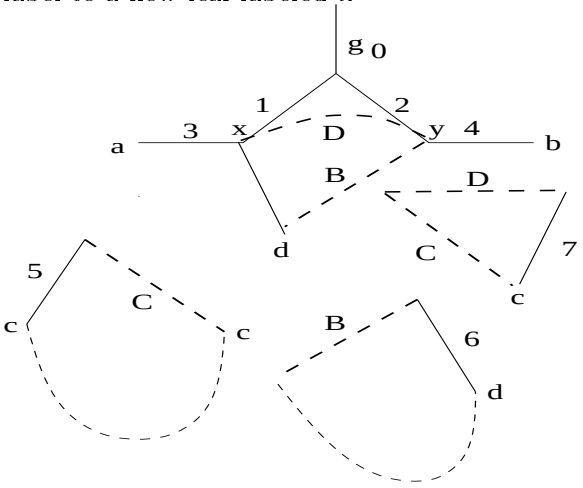
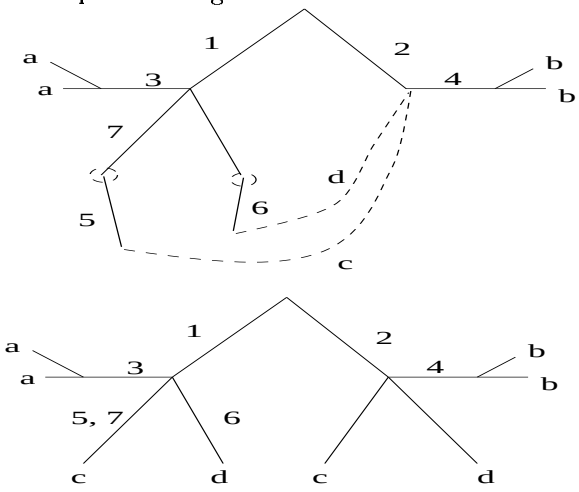


**Figure 5: The perfect phylogeny corresponding to twisting marker edges B, C, and D. The first figure shows the graph after the twistings, and the second figure shows the graph after cleanups discussed in the caption of Figure 4.**




non-trivial upper bound on the number of distinct solutions.

In a different direction, we will try to handle data where a limited amount of recombination may have occurred. The perfect phylogeny problem (with binary sequences not mixed by haplotyping) with unlimited recombination was shown to be NP-complete, but under some moderate assumptions, has a polynomial time solution [25].

Similarly, we need to deal with situations where the infinite-sites assumption is violated, but only by a small amount.

Next, there is the question of more than two possible alleles at a site. The perfect phylogeny problem for more than two states has a polynomial time solution for every fixed number of states [16]. The genotype data will specify which two states occur at each site, without partitioning the data into two haplotypes.

Finally, I believe that this is the first paper to connect the graph realization literature to a problem related to phylogeny, and I don't believe the phylogeny community is aware of these powerful techniques. Finding additional applications for these methods in phylogenetics remains a general open problem.

## 5. ACKNOWLEDGEMENTS

I would like to thank Chuck Langley for introducing me to the issue of haplotype inference, and continuing discussions on its importance.

Before I remembered the problem of recognizing graphic matroids and realized that the PPH problem could be solved by reduction to it, I tried to solve the PPH problem from scratch (and in hind-sight my ad hoc solution was close). My daughters Talia and Shira were very helpful then in generating trees from which they derived genotype data to challenge my methods, and for solving instances of the problem by their own ad hoc methods. These puzzles gave my children the first indication of what I do for a living, although they still don't understand how I can get paid for having such fun. One of their eight year-old friends found a missing case in an analysis I was doing. I thank them all for those efforts, even though the current paper does not reflect that work.

I also thank Sam Rash for early discussions and for poking holes in my first efforts. Thanks to Katherine St. John for helping to try to read part of the coalescent literature.

Research partially supported by grant DBI-9723346 from the National Science Foundation.

## 6. REFERENCES

[1] R. E. Bixby and D. K. Wagner. An almost linear-time algorithm for graph realization. *Mathematics of Operations Research*, 13:99–123, 1988.

[2] 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.

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

[4] W. H. Cunningham and J. Edmonds. A combinatorial decomposition theory. *Can. J. Math.*, 32:734–765, 1980.

[5] M. Daly, J. Rioux, S. Schaffner, T. Hudson, and E. Lander. Fine-structure haplotype map of 5q31: implications for gene-based studies and genomic ld mapping. Abstract of talk presented at the American Associate of Human Genetics National meeting, October 14, 2001

[6] 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.

[7] L. Friss, R. Hudson, A. Bartoszewicz, J. Wall, T. Donfalk, and A. Di Rienzo. Gene conversion and differential population histories may explain the contrast between polymorphism and linkage disequilibrium levels. *Am. J. of Human Genetics*, 69:831–843, 2001.

[8] 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.

[9] D. Gusfield. Efficient algorithms for inferring evolutionary history. *Networks*, 21:19–28, 1991.

[10] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

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

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

[13] J. E. Hopcroft and R.E. Tarjan. Dividing a graph into triconnected components. *SIAM J. on Computing*, 2:135–157, 1973.

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

[15] L. Jin, P. Underhill, V. Doctor, R. Davis, P. Shen, L. Luca Cavalli-Sforza, and P. Oefner. Distribution of haplotypes from a chromosome 21 region distinguishes multiple prehistoric human migrations. *Proc. of the Nat. Academy of Science*, 96:3796–3800, 1999.

[16] S. Kannan and T. Warnow. Inferring evolutionary history from DNA sequences. *SIAM J. on Computing*, 23:713–737, 1994.

[17] M.K. Kuhner and J. Felsenstein. Sampling among haplotype resolutions in a coalescent-based genealogy sampler. *Genetic Epidemiology*, 19:S15–S21, 2000.

[18] F. McMorris. On the compatibility of binary qualitative taxonomic characters. *Bull. Math. Biology*, 39:133–138, 1977.

[19] S. Orzack, D. Gusfield, and V. Stanton. Experimental and theoretical inferal of haplotypes. In preparation.

[20] I. Pe'er, R. Shamir, and R. Sharan. Incomplete directed perfect phylogeny. In D. Sankoff, editor, *Eleventh Annual Symposium on Combinatorial Pattern Matching (CPM'00)*, pages 143–153, 2000.

[21] J. C. Stephens and et. al. Haplotype variation and linkage disequilibrium in 313 human genes. *Science*, 293:489–493, 2001.

[22] 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.

[23] S. Tavare. Calibrating the clock: Using stochastic processes to measure the rate of evolution. In E. Lander and M. Waterman, editors, *Calculating the Secretes of Life*. National Academy Press, 1995.

[24] W.T. Tutte. An algorithm for determining whether a given binary matroid is graphic. *Proc. of Amer. Math. Soc*, 11:905–917, 1960.

[25] L. Wang, K. Zhang, and L. Zhang. Perfect phylogenetic networks with recombination. *J. of Comp. Biology*, 8:69–78, 2001.

[26] W. T. Whitney. Congruent graphs and the connectivity of graphs. *American Math. J.*, 54:150–168, 1932.

[27] W. T. Whitney. 2-isomorphic graphs. *American Math. J.*, 55:245–254, 1933.