Algorithms for Imperfect Phylogeny Haplotyping (IPPH) with a Single Homoplasy or Recombination Event[*]

Yun S. Song, Yufeng Wu, and Dan Gusfield

Department of Computer Science,
University of California, Davis, CA 95616, USA
yssong@cs.ucdavis.edu, wuyu@cs.ucdavis.edu,
gusfield@cs.ucdavis.edu

Abstract. The haplotype inference (HI) problem is the problem of inferring $2n$ haplotype pairs from n observed genotype vectors. This is a key problem that arises in studying genetic variation in populations, for example in the ongoing HapMap project [5]. In order to have a hope of finding the haplotypes that actually generated the observed genotypes, we must use some (implicit or explicit) genetic model of the evolution of the underlying haplotypes. The Perfect Phylogeny Haplotyping (PPH) model was introduced in 2002 [9] to reflect the "neutral coalescent" or "perfect phylogeny" model of haplotype evolution. The PPH problem (which can be solved in polynomial time) is to determine whether there is an HI solution where the inferred haplotypes can be derived on a perfect phylogeny (tree).

Since the introduction of the PPH model, several extensions and modifications of the PPH model have been examined. The most important modification, to model biological reality better, is to allow a *limited* number of biological events that violate the perfect phylogeny model. This was accomplished implicitly in [7,12] with the inclusion of several heuristics into an algorithm for the PPH problem [8]. Those heuristics are invoked when the genotype data cannot be explained with haplotypes that fit the perfect phylogeny model. In this paper, we address the issue *explicitly*, by allowing one recombination or homoplasy event in the model of haplotype evolution. We formalize the problems and provide a polynomial time solution for one problem, using an additional, empirically-supported assumption. We present a related framework for the second problem which gives a practical algorithm. We believe the second problem can be solved in polynomial time.

1 Introduction

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

[*] Research partially supported by grant EIA-0220154 from the National Science Foundation.

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.

Today, the underlying data that forms a haplotype is usually a vector of values of *m single nucleotide polymorphisms (SNP's)*. A SNP is a single nucleotide site where exactly two (of four) different nucleotides occur in a large percentage of the population. Genotype data is represented as an n by m 0-1-2 (ternary) matrix G. Each row is a genotype. A pair of binary vectors of length m (haplotypes) *generate* a row i of G if for every position c both entries in the haplotypes are 0 (or 1) if and only if $G(i,c)$ is 0 (or 1) respectively, and exactly one entry is 1 and one is 0 if and only if $G(i,c) = 2$. The international Haplotype Map Project [5] is focused on determining the common SNP haplotypes in several diverse human populations.

Given an input set of n genotype vectors of length m, the *Haplotype Inference (HI) Problem* is to find a set of n pairs of binary vectors (with values 0 and 1), one pair for each genotype vector, such that each genotype vector is generated by the associated pair of haplotypes. The ultimate goal is to computationally infer the true haplotype pairs that generated the genotypes. This would be impossible without the implicit or explicit use of some genetic model to guide the algorithm in constructing a solution. A powerful genetic model that has been used in the HI problem is the population-genetic concept of a *coalescent* [14,21].

The coalescent model of SNP haplotype evolution implies that the evolutionary history of $2n$ haplotypes, one from each of $2n$ individuals, can be displayed as a rooted tree T with $2n$ leaves, where some ancestral sequence labels the root of the tree, and where each of the m sites labels *exactly* one edge of the tree. A label i on an edge indicates the (unique) point in history where a mutation at site i occurred. Sequences evolve down the tree, starting from the ancestral sequence, changing along a branch $e = (u,v)$ by changing the state of any site that labels edge e. The state changes from what it is at u to the opposite state, recorded at v. The tree "generates" the resulting sequences that appear at its leaves. In more computer science terminology, the coalescent model says that $2n$ haplotype (binary) sequences that appear at the leaves of T are generated on a *perfect phylogeny*. See [9] for further explanation and justification of the perfect phylogeny haplotype model.

Generally, most solutions to the HI problem will not fit a perfect phylogeny, and this leads to **The Perfect Phylogeny Haplotyping (PPH) Problem**: Given an n by m matrix M that holds n genotypes from m sites, find n pairs of haplotypes that generate M on a perfect phylogeny.

It is the requirement that the haplotypes fit a perfect phylogeny, and the fact that most solutions to the HI problem will not, that enforce the coalescent model of haplotype evolution, and make it plausible that a solution to the PPH problem (when there is one) is biologically meaningful.

There are several polynomial-time solutions to the PPH problem [2,4,8,9] and a linear-time algorithm [6]. An additional empirical result that will be exploited in this paper is that when there is a PPH solution, it is highly likely that there is only a single, unique, PPH solution [3,4]. The frequency of uniqueness increases with the number of genotypes, and a surprisingly small number of genotypes is needed before the solution is highly likely to be unique.

Since the introduction of the PPH model, a central goal has been to extend the range of applicability of model by incorporating more biological complexity, yet preserving polynomial-time solvability. Biologically, the most important extension is to allow a limited amount of recombination or homoplasy (recurrent or back mutation) in the model of haplotype evolution. Homoplasy allows a site to mutate more than once, and hence allows a site to label more than one edge in the tree. As before, if a site i labels the directed edge (u, v), then the state of site i at u mutates to the opposite state, which is the state of i at v. If the two occurrences of site i are on the same path from the root of T, then the second occurrence is a "back mutation", otherwise it is a "recurrent mutation".

An *H-1 Phylogenetic Tree* T is derived from a perfect phylogeny by allowing exactly one site to label two distinct edges of T. An H-1 phylogenetic tree generates M if the sequences in M label the leaves of T.

A single-crossover recombination between two equal-length sequences P and S creates a third "recombinant" sequence of the same length consisting of a prefix of P followed by a suffix of S. The point where the recombinant sequence changes from P to S is called the "crossover" or "break" point. Recombination occurs during meiosis (and in other contexts) and is a primary mechanism creating genomic diversity in a population.

An *R-1 Phylogenetic Network* N is derived from a perfect phylogeny by allowing one recombination event, represented at a node v of N by two edges entering v, one from the node labeled by sequence P, and one labeled by sequence S. The crossover point is also noted at v. a recombination node (e.g., to have two incoming edges), where a single-crossover recombination occurs. An R-1 phylogenetic network generates M if the sequences of M label the terminal nodes (nodes with no descendants) of N. An R-1 phylogenetic network can also be described as a galled-tree with exactly one gall [11].

Given an input set of genotypes M, we define two problems.

1. The H-1 Imperfect Phylogeny Haplotyping (IPPH) Problem: Find an H-1 Phylogenetic Tree generating haplotypes that solve the HI problem for M.
2. The R-1 Imperfect Phylogeny Haplotyping (IPPH) Problem: Find an R-1 phylogenetic Network generating haplotypes that solve the HI problem for M.

In this paper, we develop algorithms for both problems. Both solutions first solve a PPH problem for a subset of the data, and we will assume (following the observations in [3,4]) that those solutions are unique. Given that assumption, our solution to the H-1 IPPH problem runs in polynomial time. We have implemented our H-1 IPPH algorithm in C++ and evaluated its performance using simulated data. As we elaborate later, our study shows that our method is both practical and highly accurate. We are currently working on extending our method to IPPH with multiple homoplasy events. Our present solution for the R-1 IPPH problem takes exponential time, but a polynomial-time algorithm for that approach looks promising, and we believe a related, more complex, method runs in polynomial time.

In what follows, we use M to denote an $n \times m$ genotype matrix. Its haplotype matrix, of size $2n \times m$, is denoted by M'. Following [2], we say that two rows in M' are *mates* if they come from a single row in M.

2 IPPH with a Single Homoplasy Event

In this section, we construct an IPPH method that allows for exactly one back or recurrent mutation. Suppose that a genotype matrix M does not admit a PPH solution. If that is due to a single homoplasy event at a particular site, then removing the column in M corresponding to that site will render the remaining data M_r compatible with a perfect phylogeny, and therefore there will be a PPH solution for M_r. In our work, we consider partitioning M column-wise into two parts, one (denoted M_s) containing exactly one column of M and the other (denoted M_r) the rest. We denote this partitioning by $M \longmapsto M_r \oplus M_s$. Our algorithm proceeds by trying all such partitions of M and checking whether M_r admits a PPH solution (i.e. a perfect phylogeny T), and, if so, whether the single column in M_s can be explained by a single homoplasy event in T, possibly after some refinement. If these conditions are satisfied by at least one partition of M, then we say that there exists a solution to IPPH with a single homoplasy event.

2.1 H-1 IPPH When There Exists a Unique PPH Solution for M_r

In what follows, we describe our main ideas through an explicit example. Consider the genotype matrix M shown on the left hand side of Figure 1. The only partition of that genotype matrix that leads to a PPH solution for M_r is the one shown on the right hand side of Figure 1. In fact, that M_r admits a unique PPH solution M'_r, shown on the left hand side of Figure 2; a PPH solution M'_s for M_s is shown there as well. The question that remains is whether we can appropriately combine M'_r with M'_s to create an H-1 IPPH solution M' for the entire genotype matrix M; i.e., for each $1 \leq i \leq n$, we want to ask whether we

M

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
g_1	1	2	1	2	1	2	2
g_2	2	2	2	0	1	0	0
g_3	2	2	2	2	1	2	2
g_4	1	1	0	1	0	0	0
g_5	1	1	1	1	2	2	0

$\longmapsto$

M_r

	i_1	i_2	i_4	i_5	i_6	i_7
g_1	1	2	2	1	2	2
g_2	2	2	0	1	0	0
g_3	2	2	2	1	2	2
g_4	1	1	1	0	0	0
g_5	1	1	1	2	2	0

$\oplus$

M_s

	i_3
g_1	1
g_2	2
g_3	2
g_4	0
g_5	1

Fig. 1. Partition of M into M_r and M_s, where M_s contains column i_3

M'_r

	i_1	i_2	i_4	i_5	i_6	i_7
r_1	1	0	0	1	0	0
r'_1	1	1	1	1	1	1
r_2	1	1	0	1	0	0
r'_2	0	0	0	1	0	0
r_3	0	0	0	1	0	0
r'_3	1	1	1	1	1	1
r_4	1	1	1	0	0	0
r'_4	1	1	1	0	0	0
r_5	1	1	1	0	0	0
r'_5	1	1	1	1	1	0

M'_s

	i_3
s_1	1
s'_1	1
s_2	0
s'_2	1
s_3	0
s'_3	1
s_4	0
s'_4	0
s_5	1
s'_5	1

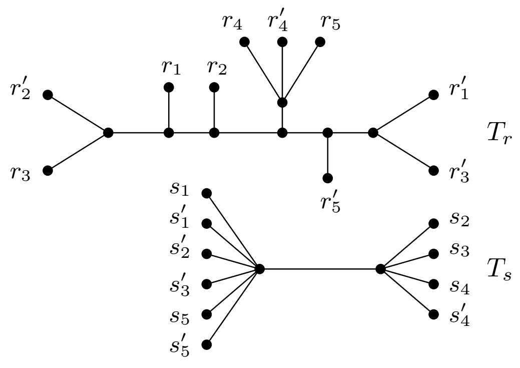


Fig. 2. Separate PPH solutions and perfect phylogenies for M_r and M_s in Figure 1

can appropriately order the rows s_i, s'_i in M'_s with respect to the rows r_i, r'_i in M'_r, such that the combined matrix is an H-1 IPPH solution for M. For each $1 \leq i \leq n$, row r_i can get paired with either s_i or s'_i. Therefore, if r_i and r'_i are distinct, and so are s_i and s'_i, for all $1 \leq i \leq n$, then there are 2^n possible ways of pairing the rows. Hence, checking whether there exists an H-1 IPPH solution for each way of pairing would be impractical when n is large. The approach we take is to work not with haplotype matrices directly but with perfect phylogenies. The problem of finding an H-1 IPPH solution therefore translates to a graph theoretical problem.

Returning to our example, consider the perfect phylogenies shown on the right hand side of Figure 2. Trees T_r and T_s correspond to the PPH solutions M'_r and M'_s, respectively. To create an H-1 IPPH solution for the entire genotype matrix M, we need to combine the information contained in T_r with that in T_s, but, before we can do that, we first need to identify the leaf labels r_i, r'_i in T_r with the leaf labels s_i, s'_i in T_s. There are $O(2^n)$ ways to do this in general. But, an important observation allows us to avoid considering all $O(2^n)$ pairings of leaf labels explicitly. Because we do not know *a priori* how the leaf labels r_i, r'_i in T_r should be paired up with the leaf labels s_i, s'_i in T_s, we can actually use that freedom to set r_i, r'_i, s_i, s'_i equal to a new label, say x_i, and study the re-labeled trees $\widetilde{T}_r$ and $\widetilde{T}_s$ to see whether there exists an H-1 IPPH solution; an H-1 IPPH solution exists if having two mutation events in $\widetilde{T}_r$ can induce the same bipartition of the *multiset* $\{x_1, x_1, x_2, x_2, \ldots, x_n, x_n\}$ as that captured by the tree topology of $\widetilde{T}_s$. Once the location of the two mutation events in $\widetilde{T}_r$ is determined, we can go back to T_r and determine the phase of the entire data M; i.e., the location of the two mutation events in T_r determines the order of s_i, s'_i in M_s with respect to r_i, r'_i in M_r.

Equivalently, if $\widetilde{T}_r$ is a binary tree, an H-1 IPPH solution exists if there exist two edges e_1 and e_2 in $\widetilde{T}_r$ that are not incident with a common vertex, such that removing those edges partitions $\widetilde{T}_r$ into three subtrees, of which two are non-adjacent, with the following properties:

(i) The multiset union of leaf labels for the two non-adjacent subtrees—i.e., those subtrees not joined by e_1 or e_2 in $\widetilde{T}_r$—is equal to the multiset of leaf labels on one side of the unique interior edge e_I in $\widetilde{T}_s$, and

(ii) the multiset of leaf labels for the remaining subtree is equal to that on the other side of e_I in $\widetilde{T}_s$.

Whether there exists such a pair of edges can easily be answered in polynomial time. If the answer is affirmative, then the phasing of the single column in M_s can be determined, up to exchange of 0s with 1s in the entire column, by looking at the topology of T_r and the position of the two mutation events in T_r.

If $\widetilde{T}_r$ is not binary, things are more complicated. For ease of discussion, we introduce the following definition (see [17] for graph theoretical terminology):

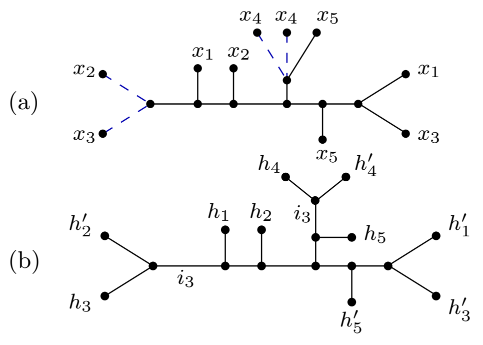
Definition 1 (Cut-subtree). *By a cut-subtree τ of a leaf-labeled unrooted tree T, we mean a subtree of T that can be obtained by removing an edge e in T, or by first refining a vertex of degree ≥ 4 and then removing the newly created edge e. The remaining part of T, after deleting any degree-2 vertex created by removing e, is denoted $T \setminus \tau$.*

If $\widetilde{T}_r$ is not binary, we need to ask whether there exist two disjoint *cut-subtrees* τ_1 and τ_2 of $\widetilde{T}_r$, such that

(i′) the multiset union of leaf labels for τ_1 and τ_2 is equal to that for one side of e_I in $\widetilde{T}_s$, and

(ii′) the remaining part of T_r has a label multiset equal to that for the other side of e_I in $\widetilde{T}_s$.

A polynomial-time algorithm for finding such cut-subtrees, if they exist, is described in Section 2.2. For now, we return to the simple example in Figure 2. It



(c)

M'

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
h_1	1	0	1	0	1	0	0
h_1'	1	1	1	1	1	1	1
h_2	1	1	1	0	1	0	0
h_2'	0	0	0	0	1	0	0
h_3	0	0	0	0	1	0	0
h_3'	1	1	1	1	1	1	1
h_4	1	1	0	1	0	0	0
h_4'	1	1	0	1	0	0	0
h_5	1	1	1	1	0	0	0
h_5'	1	1	1	1	1	1	0

Fig. 3. (a) Cut-subtrees, denoted by dashed lines, in $\widetilde{T}_r$ that satisfy properties (i′) and (ii′). (b) Imperfect phylogenetic tree with two mutations for column i_3. The two edges on which mutations for column i_3 occur are labeled i_3. (c) The corresponding H-1 IPPH solution for the entire genotype matrix M.

is easy to see that there does not exist a pair of edges in $\widetilde{T}_r$ such that properties (i) and (ii) shown above are both satisfied. However, there exist two cut-subtrees, shown in Figure 3a, that satisfy properties (i′) and (ii′). Having identified the appropriate cut-subtrees, we can now go back to the original tree T_r and determine the phase of M_s. An imperfect phylogenetic tree with two mutations for column i_3 and the corresponding H-1 IPPH solution are shown in Figures 3b and 3c, respectively, where haplotype mates are now labeled h_i and h'_i. Note that the H-1 IPPH solution is unique. In terms of pairing the rows in M'_r and M'_s, the H-1 IPPH solution corresponds to pairing s_2 with r'_2 (and hence s'_2 with r_2) and s_3 with r_3 (and hence s'_3 with r'_3). In this example, the imperfect phylogeny for M' is binary. In general, an imperfect phylogeny for M' may still contain vertices of degree greater than 3.

2.2 Algorithm for Finding Appropriate Cut-Subtrees in Non-binary Trees for a Single Homoplasy Event

Suppose that $\widetilde{T}_r$ is non-binary. In what follows, the reader should refer to Figure 4 for illustration of notation. If we remove two existing edges E_1 and E_2 from $\widetilde{T}_r$, then that defines two non-adjacent subtrees T_1 and T_2. Our goal is to check whether we can choose a set of edges from $e_1, \ldots, e_p$ to create a cut-subtree τ_1 and, similarly, choose a set of edges from $e_{p+1}, \ldots, e_q$ to create a cut-subtree τ_2, such that properties (i′) and (ii′) in Section 2.1 are satisfied. Below we provide a polynomial-time algorithm for finding all possible such pairs of disjoint cut-subtrees, when they exist.

Our algorithm is based on coloring a graph G whose q vertices $v_1, \ldots, v_q$ are in one-to-one correspondence with $e_1, \ldots, e_q$; v_i in G is related to e_i in $\widetilde{T}_r$. Edges in G will be defined shortly. A coloring procedure may terminate before reaching the end, if inconsistency is encountered; i.e., if a vertex is assigned more than one color. If G is colored consistently, the final coloring of the vertices in G determines which of $e_1, \ldots, e_q$ should be chosen to construct the desired cut-subtrees τ_1 and τ_2. In our convention, if vertex v_i is colored "red," then it means we should "take" e_i. If it is colored "black," then we should "not take" e_i.

We use $\mathcal{L}(T)$ to denote the set of leaf labels in T. More generally, $\mathcal{L}(T)$ is a multiset when we consider trees with duplicate labels. Let $X|Y$ be the bipartition of $\mathcal{L}(\widetilde{T}_s)$ defined by the single interior edge in $\widetilde{T}_s$, and recall that $\widetilde{T}_r$ and $\widetilde{T}_s$ carry duplicate leaf labels. Our algorithm is as follows.

1. Choose a pair of existing edges E_1, E_2 in $\widetilde{T}_r$ that has not been tried so far. If no such choice remains, terminate the algorithm.
2. Remove E_1, E_2 to partition $\widetilde{T}_r$ into three subtrees. Let T_1, T_2 be the two non-adjacent subtrees as depicted in Figure 4. If $\mathcal{L}(T_1) \cup \mathcal{L}(T_2)$ contains either X or Y, or both, create q vertices $v_1, \ldots, v_q$, and go to next step. If not, go back to step 1; no solution is possible for the current choice of E_1, E_2.
3. For each $Z \in \{X, Y\}$ satisfying $Z \subset \mathcal{L}(T_1) \cup \mathcal{L}(T_2)$, check whether the following conditions hold:

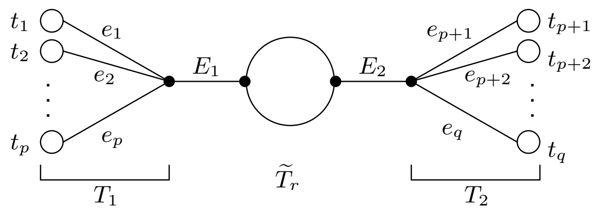


Fig. 4. Schematic depiction of $\widetilde{T}_r$. Here, t_i denote subtrees and the big circle in the center schematically represents the rest of $\widetilde{T}_r$.

   (a) If x appears only once in Z, then there does not exist a t_k in T_1 or T_2 such that $\mathcal{L}(t_k)$ contains two xs.
   (b) For every x that appears exactly once in Z, create an edge between v_i and v_j if $x \in \mathcal{L}(t_i)$ and $x \in \mathcal{L}(t_j)$. Let G_Z denote the resulting graph. Then, every non-trivial connected component of G_Z is bipartite[1].

   If no Z satisfies the above conditions, go back to step 1; there is no solution for the current choice of E_1, E_2. Otherwise, pass G_Z to the next step.
4. For each G_Z passed, check whether it is possible to color the vertices in G_Z as describe below without encountering inconsistency.
   (a) If $x \notin Z$, find all t_k such that $x \in \mathcal{L}(t_k)$ and color v_k black.
   (b) If x occurs twice in Z, find all t_k such that $x \in \mathcal{L}(t_k)$ and color v_k red.
   (c) If x occurs only once in Z and there exists exactly one t_k such that $\mathcal{L}(t_k)$ contains a single x, color v_k red.

   If no G_Z admits consistent coloring, go back to step 1. Otherwise, pass consistently colored G_Z to the next step.
5. For each non-trivial connected component of G_Z, see whether any of the vertices has been colored. If so, color the remaining uncolored vertices, if there are any, in that connected component to respect the bipartite structure (i.e. red on one side and black on the other). If this is not possible, return to step 1.
6. If there are k totally-uncolored non-trivial connected components of G_Z, then there are 2^k ways to color them consistently, and hence there are 2^k solutions for the current choice of E_1, E_2. Go back to step 1.

The above algorithm can be implemented to run in $O(n^4)$ time, where $2n$ is the number of leaves.

2.3 Empirical Results

We implemented our H-1 IPPH algorithm in C++ and compared its performance on simulated data with that of PHASE [20]. The input datasets were generated as follows. We first used Hudson's program MS [15] to generate homoplasy-free haplotype datasets satisfying the 5% rule[2] described in [4]. To introduce

[1] If a non-trivial connected component is not bipartite, then it has an odd-length cycle, which leads to inconsistent coloring.

[2] The 5% rule is a biologically relevant restriction that every column in the haplotype matrix has minor allele frequency $\geq 5\%$.

Table 1. Comparison of our H-1 IPPH method with PHASE for genotype matrices of size $n \times m$. Shown here are average accuracy measures and average running time (on a 2.8 GHz Pentium PC) per dataset, based on 100 datasets of each size. Our method seems comparable to PHASE in accuracy, while being significantly faster than PHASE.

	Our H-1 IPPH method				PHASE			
	50×50	100×50	50×100	100×100	50×50	100×50	50×100	100×100
Standard error	0.005	0.005	0.006	0.003	0.006	0.002	0.007	0.001
Switch accuracy	0.999	0.999	1.000	1.000	0.998	0.999	0.999	1.000
% of misphased 2s	0.03%	0.03%	0.02%	0.01%	0.07%	0.02%	0.03%	0.01%
Running time	0.22s	0.41s	1.52s	3.09s	14.2s	27.7s	43.6s	85.8s

Table 2. Performance of our H-1 IPPH method in more detail. The number of datasets shown is out of 100.

	50×50	100×50	50×100	100×100
# of datasets admitting PPH solutions	20	19	16	15
# of datasets admitting H-1 IPPH solutions (with a unique PPH solution for M_r)	80 (80)	81 (81)	84 (84)	85 (85)
Frequency of correctly identifying the homoplasy column when M admits no PPH solution	95%	98%	96%	98%

a homoplasy event, we randomly chose two distinct edges (on which mutations occur) in the underlying rooted genealogical tree of each haplotype dataset, with the probability of choosing an edge being proportional to its length. This process was repeated for each dataset until a homoplasy site satisfying the 5% rule got generated. We then randomly inserted the so obtained homoplasy column into the original haplotype matrix. Finally, a genotype matrix was created by pairing row $2i$ with row $2i - 1$ in the modified haplotype matrix.

Let G_M denote the set of genotypes in dataset M with more than one heterozygous site. We used the following three measures of haplotype reconstruction accuracy: (a) The *standard error* [20] is the ratio of the number of genotypes in G_M whose haplotypes are incorrectly inferred to the total number of genotypes in G_M. (b) For a genotype g in G_M, the *switch accuracy* [16] of its inferred haplotypes is defined as $(h - w - 1)/(h - 1)$, where h is the number of heterozygous sites in g and w is the number of switches between neighboring heterozygous sites needed to transform the inferred haplotypes to the true haplotypes. The switch accuracy averaged over the genotypes in G_M defines the switch accuracy of the entire inferred haplotype matrix M'. (c) The last measure is the percentage of misphased 2s with respect to the total number of 2s in G_M.

Our simulation results are summarized in Table 1. For each size $n \times m$, we used 100 simulated genotype matrices. As the table shows, our method is comparable to PHASE in terms of accuracy, while being tens of times faster than PHASE. As shown in Table 2, for each combination of n and m used, more than 80 out of 100 datasets did not admit PPH solutions. Every such a dataset had an

H-1 IPPH solution with a *unique* PPH solution for the genotype matrix M_r, in agreement with our assumption. Also, note that, for datasets with no PPH solutions, our H-1 IPPH method correctly identified the homoplasy column (a feature that PHASE does not have) with very high accuracy. This study shows that our method is both practical and highly accurate.

3 IPPH with a Single Recombination Event

The case with exactly one single-crossover recombination event is similar in spirit to the case of a single homoplasy event. Suppose that an $n \times m$ genotype matrix M does not admit a PPH solution. If that is due to a single recombination event with a breakpoint[3] b somewhere between 1 and m, then the part to the left of b and that to the right of b should each admit a PPH solution. In our approach, we choose a recombination breakpoint b somewhere between 1 and m, and consider partitioning M column-wise into two parts, one containing the columns to the left of b and the other the columns to the right of b. We denote a partitioning by $M \longmapsto M_L \oplus M_R$, with L (resp. R) denoting left (resp. right). Our algorithm proceeds by trying all such partitions of M and checking whether each of M_L and M_R admits a PPH solution, and, if so, whether the PPH solutions from the two parts can be combined in a way consistent with there being a single recombination event, i.e. a galled tree with one gall [11,10]. If these conditions are satisfied by at least one partition of M, then we say that there exists a solution to IPPH with a single recombination event.

3.1 R-1 IPPH When There Exist Unique PPH Solutions for Each Side

Given PPH solutions M'_L and M'_R for M_L and M_R, respectively, the main question that we need to ask is whether we can appropriately pair up the mates L_i, L'_i in M'_L with the mates R_i, R'_i in M'_R, such that there is a galled tree with one gall for the combined data. If there are n genotypes in M, then in the worst case there are 2^n ways of doing the pairing. Similar to what we discussed in Section 2.1 for the case of a single homoplasy event, we propose to solve this problem in the following way: First, for all $1 \leq i \leq n$, we set $L_i = L'_i = R_i = R'_i = x_i$, where x_i is a new leaf label. Then, we work with perfect phylogenies, not with haplotype matrices, as described below.

Let $\widetilde{T}_L$ and $\widetilde{T}_R$ denote the re-labeled perfect phylogenies corresponding to M'_L and M'_R, respectively. Then, what properties of $\widetilde{T}_L$ and $\widetilde{T}_R$ would imply that there exists an R-1 IPPH solution? In a galled tree with exactly one gall, whose recombination breakpoint is denoted b, the rooted tree τ_L to the left of b and the rooted tree τ_R to the right of b are closely related. More precisely, there exists a single tree rearrangement operation, called the subtree-prune-and-regraft (SPR) operation, that one can perform on τ_L to transform it into τ_R, or vice versa [1,13,18,19]. This implies that $\widetilde{T}_L$ and $\widetilde{T}_R$ cannot be two arbitrary

[3] A breakpoint occurs between two sites.

trees for there to be an R-1 IPPH solution. Rather, they, too, should be related by an SPR-like operation. There is an important point that we should highlight here. For SPR operations to be biologically meaningful, certain restrictions must be imposed to avoid possible contradictions [18,19]. For example, time-ordering of certain associated biological events must be obeyed. The reader should be warned that, as there exists no fixed sense of time direction in unrooted trees, performing SPR operations on them and drawing conclusions about evolutionary histories, of which time is an essential component, may not be the right thing to do. However, because we are working with the case of a single recombination event, which involves a single SPR operation, it is always possible to root the two unrooted trees involved to construct a consistent evolutionary history. To recapitulate, there exists an R-1 IPPH solution if $\widetilde{T}_L$ and $\widetilde{T}_R$ are related by a single SPR operation.

Determining whether two binary trees $\widetilde{T}_L$ and $\widetilde{T}_R$ are exactly one SPR operation away is equivalent to checking whether there exists a common cut-subtree t in $\widetilde{T}_L$ and $\widetilde{T}_R$ such that $\widetilde{T}_L \setminus t$ is identical to $\widetilde{T}_R \setminus t$. It is straightforward to do this in polynomial time. It is important to note that, in general, perfect phylogenies that we need to compare may not be binary. When either one or both of $\widetilde{T}_L$ and $\widetilde{T}_R$ are non-binary, to determine whether they are exactly one SPR operation way, we need to check whether there exist a cut-subtree t_L of $\widetilde{T}_L$ and a compatible[4] cut-subtree t_R of $\widetilde{T}_R$, such that $\widetilde{T}_L \setminus t_L$ is compatible with $\widetilde{T}_R \setminus t_R$. A brute-force way of checking the 1-SPR condition is as follows. As shown in Figure 4, remove two edges E_1 and E_2 from $\widetilde{T}_L$ to obtain two subtrees T_1 and T_2. We need to find two cut-subtrees $\tau_1 \subseteq T_1$ and $\tau_2 \subseteq T_2$ so that we can prune τ_1 and regraft it next to τ_2, or vice versa, and check whether the resulting tree $\widetilde{T}'_L$ is compatible with $\widetilde{T}_R$. To do this, simply enumerate all possible ways of generating cut-subtrees τ_1 and τ_2. After pruning and regrafting, test for compatibility. This simple method is feasible when there are not too many unrefined vertices of large degree. We believe that there is a polynomial-time algorithm for checking the 1-SPR condition, as well as a related, more complex, method for solving the R-1 IPPH problem that runs in polynomial time.

3.2 An Example

Consider the genotype matrix shown on the left hand side of Figure 5. For every partition $M \mapsto M_L \oplus M_R$, we first need to check whether there exist PPH solutions for M_L and M_R. For the particular partition shown on the right hand side of Figure 5, there exists a unique PPH solution for each of M_L and M_R. These PPH solutions M'_L and M'_R, and the corresponding the perfect phylogenies T_L and T_R, respectively, are shown in Figure 6. After redefining $L_i = L'_i = R_i = R'_i = x_i$, we obtain the trees $\widetilde{T}_L$ and $\widetilde{T}_R$ shown in Figure 7. Now, note that, $\widetilde{T}_L$ and $\widetilde{T}_R$ contain a common cut-subtree t such that $\widetilde{T}_L \setminus t$ is identical to $\widetilde{T}_R \setminus t$. Hence, $\widetilde{T}_L$ and $\widetilde{T}_R$ can be related by a single SPR operation in which t gets

[4] Two trees T_1, T_2 are said to be compatible if there exists a third tree that is a refinement of both T_1 and T_2.

M

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
g_1	0	2	1	2	1	1	1	2	2	2
g_2	2	2	1	1	2	1	2	2	1	1
g_3	1	1	2	1	2	2	2	1	2	2
g_4	2	1	2	2	1	2	1	1	2	0

$\longmapsto$

M_L

	i_1	i_2	i_3	i_4	i_5
g_1	0	2	1	2	1
g_2	2	2	1	1	2
g_3	1	1	2	1	2
g_4	2	1	2	2	1

$\oplus$

M_R

	i_6	i_7	i_8	i_9	i_{10}
g_1	1	1	2	2	2
g_2	1	2	2	1	1
g_3	2	2	1	2	2
g_4	2	1	1	2	0

Fig. 5. A partition of M into M_L and M_R

M'_L

	i_1	i_2	i_3	i_4	i_5
L_1	0	1	1	0	1
L'_1	0	0	1	1	1
L_2	0	0	1	1	1
L'_2	1	1	1	1	0
L_3	1	1	0	1	1
L'_3	1	1	1	1	0
L_4	0	1	1	0	1
L'_4	1	1	0	1	1

M'_R

	i_6	i_7	i_8	i_9	i_{10}
R_1	1	1	0	1	1
R'_1	1	1	1	0	0
R_2	1	1	0	1	1
R'_2	1	0	1	1	1
R_3	1	0	1	1	1
R'_3	0	1	1	0	0
R_4	1	1	1	1	0
R'_4	0	1	1	0	0

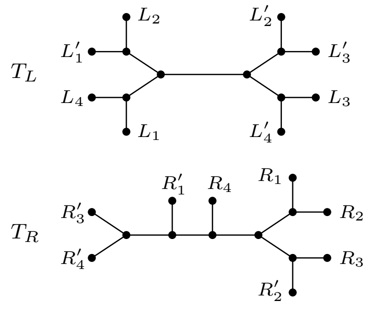
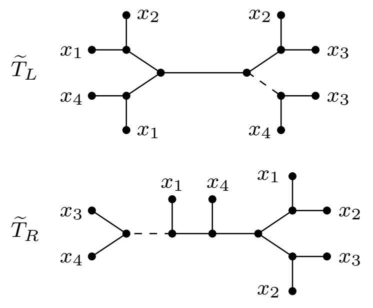


Fig. 6. Separate PPH solutions for M_L and M_R in Figure 5, and their corresponding perfect phylogenies



M'

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
h_1	0	1	1	0	1	1	1	1	0	0
h'_1	0	0	1	1	1	1	1	0	1	1
h_2	0	0	1	1	1	1	1	0	1	1
h'_2	1	1	1	1	0	1	0	1	1	1
h_3	1	1	0	1	1	0	1	1	0	0
h'_3	1	1	1	1	0	1	0	1	1	1
h_4	0	1	1	0	1	1	1	1	1	0
h'_4	1	1	0	1	1	0	1	1	0	0

Fig. 7. Re-labeled perfect phylogenies and an R-1 IPPH solution for the entire genotype matrix M in Figure 5. If the edges denoted by dashed lines are removed, then a common 2-leaved subtree t labeled by x_3 and x_4 gets pruned, and $\widetilde{T}_L \setminus t$ and $\widetilde{T}_R \setminus t$ become identical.

pruned and regrafted to transform $\widetilde{T}_L$ into $\widetilde{T}_R$, or vice versa. Going back to T_L and T_R with original leaf labels, it is then possible to conclude that L_1 should get paired with R'_1, L_2 with R_2, L_3 with R'_3, and L_4 with R_4. The R-1 IPPH solution just described is shown on the right hand side of Figure 7, where h_i and h'_i denote haplotype mates for genotype g_i.

The partition shown in Figure 5 led to two binary trees. Other partitions do not have this nice property, however. For instance, the partition that divides M between columns 4 and 5 leads to a non-binary tree $\widetilde{T}_L$ for the left part and a binary tree $\widetilde{T}_R$ for the right part. There are several other possible partitions such that each part admits a PPH solution; but, in fact, all such partitions lead to the same R-1 IPPH solution shown in Figure 7.

References

1. B. L. Allen and M. Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Ann. Combin.*, 5:1–13, 2001.
2. V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph. Haplotyping as perfect phylogeny: A direct approach. *J. Comput. Biol.*, 10:323–340, 2003.
3. T. Barzuza, J.S. Beckman, R. Shamir, and I. Pe'er. Computational problems in perfect phylogeny haplotyping: XOR genotypes and tag SNPs. In *Proc. of CPM*, pages 14–31, 2004.
4. R.H. Chung and D. Gusfield. Empirical exploration of perfect phylogeny haplotyping and haplotypers. In *Proc. of COCOON*, pages 5–19, 2003.
5. International HapMap Consortium. The HapMap project. *Nature*, 426:789–796, 2003.
6. Z. Ding, V. Filkov, and D. Gusfield. A linear-time algorithm for the perfect phylogeny haplotyping problem. In *Proc. of RECOMB*, pages 585–600, 2005.
7. E. Eskin, E. Halperin, and R. Karp. Large scale reconstruction of haplotypes from genotype data. In *Proc. of RECOMB*, pages 104–113, 2003.
8. E. Eskin, E. Halperin, and R.M. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. *J. Bioinf. Comput. Biol.*, 1:1–20, 2003.
9. D. Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions (Extended Abstract). In *Proc. of RECOMB*, pages 166–175, 2002.
10. D. Gusfield. Optimal, efficient reconstruction of Root-Unknown phylogenetic networks with constrained recombination. *J. Comput. Sys. Sci.*, 70:381–398, 2005.
11. D. Gusfield, S. Eddhu, and C. Langley. Optimal, efficient reconstruction of phylogenetic networks with constrained recombination. *J. Bioinf. Comput. Biol.*, 2(1):173–213, 2004.
12. E. Halperin and E. Eskin. Haplotype reconstruction from genotype data using Imperfect Phylogeny. *Bioinformatics*, 20:1842–1849, 2004.
13. J. Hein. Reconstructing evolution of sequences subject to recombination using parsimony. *Math. Biosci.*, 98:185–200, 1990.
14. R. Hudson. Gene genealogies and the coalescent process. *Oxford Survey of Evolutionary Biology*, 7:1–44, 1990.
15. R. Hudson. Generating samples under the Wright-Fisher neutral model of genetic variation. *Bioinformatics*, 18:337–338, 2002.
16. S. Lin, D.J. Cutler, M.E. Zwick, and A. Chakravarti. Haplotype inference in random population samples. *Am. J. Hum. Genet.*, 71:1129-1137, 2002.
17. C. Semple and M. Steel. *Phylogenetics.* Oxford University Press, UK, 2003.
18. Y. S. Song. On the combinatorics of rooted binary phylogenetic trees. *Ann. Combin.*, 7:365–379, 2003.
19. Y. S. Song and J. Hein. Constructing minimal ancestral recombination graphs. *J. Comput. Biol.*, 12:147–169, 2005.
20. M. Stephens, N.J. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction from population data. *Am. J. Hum. Genet.* 68:978–989, 2001.
21. S. Tavaré. Calibrating the clock: Using stochastic processes to measure the rate of evolution. In E. Lander and M. Waterman, editors, *Calculating the Secrets of Life.* National Academy Press, 1995.