Algorithm for haplotype inferring via galled-tree networks with simple galls

Arvind Gupta[1] Ján Maňuch[1] Ladislav Stacho[2]
Xiaohong Zhao[3]
[1]Department of Computer Science, University of British Columbia, Canada
[2]Department of Mathematics, Simon Fraser University, Canada
[3]Siemens Corporate Research, Princeton, USA

Abstract

The problem of determining haplotypes from genotypes has gained considerable prominence in the research community. Here the focus is on determining sets of SNP values on individual chromosomes since such information captures the genetic causes of diseases. The most efficient algorithmic tool for haplotyping is based on perfect phylogenetic trees. A drawback of this method is that it cannot be applied in situations when the data contains homoplasies (multiple mutations of the same character) or recombinations. Recently, Song et al. [21] studied the two cases: haplotyping via imperfect phylogenies with a single homoplasy and via galled-tree networks with one gall. In [6], we have shown that the haplotyping via galled-tree networks is NP-hard, even if we restrict to the case when every gall contains at most 4 mutations. We present a polynomial algorithm for haplotyping via galled-tree networks with simple galls (each having two mutations). In the end, we give the experimental results comparing our algorithm with PHASE on simulated data.

1 Introduction

With the great progress of the Human Genome project, recent research has focused on the problem of determining certain genome variants, SNPs, on individual chromosomes (*haplotypes*). Such information captures genetic variations and is already playing a central role in helping to determine the genetic causes of diseases and in designing effective pharmaceutical responses to these diseases ([3, 19]). Experimental methods allow for cost-effective determination of genotype information (the combined information of the two haplotypes for an individual across both matching chromosomes) and so the problem reduces to determining haplotypes from genotypes, the haplotyping problem. Each SNP (single nucleotide polymorphism) can take two different values over all individuals. Therefore, haplotypes can be represented with binary data, e.g., 0 representing the most common SNP value and 1 the other value. For genotypes, we use a 0 (respectively, 1) to represent that both sites are 0's (respectively, 1's), and a 2 to represent that the sites have different values (one is 0 and the other 1).

While in-vitro techniques can be used for haplotyping, the cost is prohibitive [18] and there is strong interest in the development of algorithmic tools (see [2, 9, 13] for survey of the problem). The most viable way for computing haplotypes is to use genotype data of a population of individuals. Various types of parsimonious criteria are used to choose the most plausible inference from the whole set of genotypes, including the maximum resolution problem of Clark, pure parsimony criteria, haplotyping via perfect phylogeny and several statistical methods, cf. [12] for an overview.

Gusfield [8] developed the first such exact algorithms and based these on an underlying assumption of perfect phylogeny tree on SNP sequences. In particular, each SNP site mutates at most once and there are no recombinations allowed. This is based on experimental results showing that many chromosomes are blocky and nucleotides on each block tends to inherit together ([3, 19]). As such these experiments do not exclude recombinations within a block and models were needed that allow for a few recombinations.

One possible approach to deal with recombinations is using the imperfect phylogenies that allow homoplasies. The goal is to compute haplotypes that can be arranged into phylogenetic tree with the smallest number of repeated mutations (homoplasies). While the complexity of this problem is unknown, it can be solved in polynomial time if the number of repeated mutations is assumed to be constant [22]. However, even a single recombination can introduce a large number of homoplasies in the phylogenetic tree. Another approach is to use extensions of perfect phylogenies which directly model recombinations event.

A phylogenetic network is a generalization of phylogenetic trees that in addition to mutations models also recombinations. The problem of building a phylogenetic network with the minimum number of recombinations for a set of taxa is intractable even for taxa with binary characters. Hein is among the first to give heuristic algorithms for building phylogenetic networks ([14, 15, 20]). An exact polynomial algorithm ([10]) was developed for a simplified model — the galled-tree network ([25, 10]).

The problem of utilizing galled-tree network model to explain evolutionary history for the original haplotypes that form genotypes was introduced by Gusfield et al. [11]. In [7, 6], we have shown that this problem is NP-complete, and it remains so even in the case when every gall is required to contain at most 4 mutations[1]. Consequently, even restricted versions of the problem are interesting to explore. Song et al. [21] designed a practical algorithm for haplotyping via galled-tree networks with one gall. In this paper, building on the characterization of the existence of galled-tree networks presented in [4], we present a polynomial time algorithm (with complexity $O(n^2 + nm^2)$) for haplotyping via galled-tree networks with galls having exactly two mutations, called simple galls. In addition, we require the genotype matrices to satisfy a natural condition which is implied, for example, by presence of at least one 1 in every column which contains 2. The examination of real biological genotype data and some simulation data suggests that this condition is almost always satisfied. Note that the problem of inferring haplotypes using galled-tree network generalizes the haplotyping problem via perfect phylogeny.

An extended abstract of this paper appeared in [5].

2 Definitions

2.1 Haplotype inferring from population data

Single Nucleotide Polymorphisms (SNPs) are the most frequent form of human genetic variations. A set of SNP values on a single chromosome is called a *haplotype* (e.g., SNPs that sit on a gene). Among most of the human population, SNPs take two values. The value that appears most often is called the *major allele*, the other the *minor allele*. Therefore, haplotypes are commonly represented as sequences of 0 and 1, by fixing a mapping of $\{0, 1\}$ to two possible states in $\{A, C, G, T\}$ at each SNP position. The combined information from two haplotypes for a matching pair of chromosomes is called a *genotype*. In a genotype, the information about which value comes from the first chro-

[1]Our recent unpublished result indicate that the problem remains NP-complete even for galled-trees with galls with at most 3 mutation edges.

mosome and which from the second chromosome of the matching pair is lost. A genotype sequence is usually represented as a sequence of $\{0,1,2\}$, where value 0 (1) at certain position represents the fact that both haplotypes have value 0 (1) at this position (*homozygous*), while value 2 means that the values on two haplotypes at this position differ (*heterozygous*). However, in the latter case, it is not known which of the two has value 0 and which value 1. The *haplotype inferring* problem, or simply *haplotyping*, is the problem of determining haplotype sequences from genotype sequences of a set of individuals. In this paper, we consider population data with no pedigree information known. The input is a *genotype matrix* whose rows are genotype sequences over $\{0,1,2\}$. Each row represents a genotype of one individual, and each column corresponds to the values of one SNP over all individuals. Formally, the problem can be defined as follows.

Definition 1 (Haplotypes inferring from genotypes)**.** Given a genotype $n \times m$ matrix A with values $\{0,1,2\}$, we say that a $2n \times m$ haplotype matrix B with values $\{0,1\}$ is *inferred* from A if and only if for every SNP $c \in \{1,\ldots,m\}$,

- if $A(i,c) \in \{0,1\}$, then $B(2i-1,c) = B(2i,c) = A(i,c)$; and
- if $A(i,c) = 2$, $B(2i-1,c) \neq B(2i,c)$.

Obviously, for any row, there are exponential (in the number of 2's in the row) ways to infer two haplotypes from the row.

Let $\mathcal{X}_A = \{x_{r\{c_1,c_2\}};\ A(r,c_1) = A(r,c_2) = 2\}$ be the set of Boolean variables. For brevity, we will abuse notation and refer to variable $x_{r\{c_1,c_2\}}$ by $x_{rc_1c_2}$. The value of the variable $x_{rc_1c_2}$ determines the way how the pair of 2's in the row r and columns c_1 and c_2 is resolved. Define assignment $I_B : \mathcal{X}_A \to \{0,1\}$ as follows. Let $I_B(x_{rc_1c_2}) = 0$ if the pair is resolved equally in B, i.e, $\begin{pmatrix}2 & 2\end{pmatrix} \to \begin{pmatrix}0 & 0\\ 1 & 1\end{pmatrix}$; and $I_B(x_{rc_1c_2}) = 1$ if the pair is resolved unequally in B, i.e, $\begin{pmatrix}2 & 2\end{pmatrix} \to \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$. Note that specifying the assignment I_B is equivalent to specifying the matrix B (up to swapping rows $2i-1$ and $2i$, for any $i \in \{1,\ldots,n\}$).

Given a genotype matrix A, for every $x,y \in \{0,1\}$, we say that a pair of columns c_1, c_2 *induces* $[x,y]$ *in* A, if $A[c_1,c_2]$ contains at least one of the pairs $[x,y]$, $[2,y]$ and $[x,2]$. We can define a conflict graph for genotype matrix A similarly as for haplotype matrices by relaxing the notion of conflict. In particular, character c_1 and c_2 conflict if they induce $[0,1]$, $[1,0]$ and $[1,1]$ in A.

Using the characterization for the PPH problem by Bafna et al. [1], we have the following lemma about the relationship between the conflicts B and the assignment I_B.

Lemma 1. *Given a genotype matrix A, infer a matrix B. For every $x_{rc_1c_2}$, $x_{rc_1c_3}$, $x_{rc_2c_3} \in \mathcal{X}_A$, $I_B(x_{rc_1c_2}) + I_B(x_{rc_1c_3}) + I_B(x_{rc_2c_3}) = 0$. In addition, characters c_1, c_2 conflict in B if and only if at least one of the following is true:*

(C0) c_1, c_2 induce all three pairs: $[1,1]$, $[0,1]$ and $[1,0]$ in A;

(C1) $I_B(x_{rc_1c_2}) \neq I_B(x_{r'c_1c_2})$ for some $x_{rc_1c_2}, x_{r'c_1c_2} \in \mathcal{X}_A$;

(C2) c_1, c_2 induce $[1,1]$ in A, and there is $x_{rc_1c_2} \in \mathcal{X}_A$ such that $I_B(x_{rc_1c_2}) = 1$;

(C3) c_1, c_2 induce $[0,1]$ and $[1,0]$, and there is $x_{rc_1c_2} \in \mathcal{X}_A$ with $I_B(x_{rc_1c_2}) = 0$.

2.2 Phylogenetic and galled-tree networks

We start with a definition of perfect phylogenies defined on binary characters (e.g., SNPs) with states 0 and 1. In this paper, we assume that for every character both states appear in some input sequence (other characters can easily be filtered from the input as they do not affect the solution).

Definition 2 (Perfect phylogenetic tree)**.** A *perfect phylogenetic tree* T on m characters is a rooted tree such that each vertex is labeled with a binary sequence of length m and each edge is labeled with a number from 1 to m representing the only position in which the two sequences of the end vertices of the edge differ. In addition, each number $1, \ldots, m$ is assigned to exactly one edge.

Given a set of n binary sequences, the *perfect phylogeny problem* asks whether there exists a perfect phylogenetic tree containing all the sequences. If such a tree exists we say that the set of sequences can be *explained* by a perfect phylogenetic tree.

Note that a phylogenetic tree can be defined also for characters with more than two states. The next definition extends the previous definition by allowing recombination events.

Definition 3 (Phylogenetic network)**.** A *phylogenetic network* N on m characters (usually, SNPs) is a directed acyclic graph containing exactly one vertex (the root) with no incoming edges, a set of internal vertices that have both incoming and outgoing edges, and exactly n vertices (the leaves) with no outgoing edges. Each vertex other than the root has either one or two incoming edges. If it has one incoming edge, the edge is called a *mutation edge*, otherwise it is called a *recombination edge*. A vertex x with two incoming edges is called a *recombination vertex*.

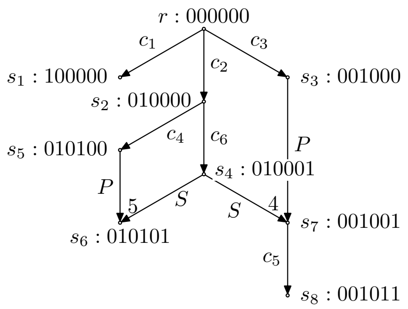
Each integer (character) from 1 to m is assigned to exactly one mutation edge in N and each mutation edge is assigned one character. Each vertex in N is labeled by a binary sequence of length m, starting with the root vertex which is labeled with the all-0 sequence. Since N is acyclic, the vertices in N can be topologically sorted into a list, where every vertex occurs in the list only after its parent(s). Using that list, we can define the labels of the non-root vertices, in order of their appearance in the list, as follows:

- For a non-recombination vertex v, let e be the mutation edge labeled c coming into v. The label of v is obtained from the label of v's parent by changing the value at position c from 0 to 1.

- Each recombination vertex x is associated with an integer $\mathrm{r}_x \in \{2, \ldots, m\}$, called the *recombination point* for x. Label the two recombination edges coming to x by P and S, respectively. Let $P(x)$ ($S(x)$) be the sequence of the parent of x on the edge labeled P (S). Then the label of x consists of the first $\mathrm{r}_x - 1$ characters of $P(x)$, followed by the last $m - \mathrm{r}_x + 1$ characters of $S(x)$. Hence $P(x)$ contributes a prefix and $S(x)$ contributes a suffix to x's sequence.

In this paper, the sequence at the root of the phylogenetic network is always the all-0 sequence, and all results are relative to that assumption. More general phylogenetic networks with an unknown root were studied in a recent paper by Gusfield [10]. Note also that our definition of phylogenetic networks differs slightly from the original definition of Wang et al. [25]. We assume that each mutation edge has exactly one label. Every phylogenetic network without this assumption can easily be transformed to our model by replacing every mutation edge with multiple labels by a sequence of edges each having one of these labels. Furthermore, all mutation edges without a label are contracted out. Our definition results in more uniform phylogenetic networks, however we cannot require that all sequences of an input matrix appear at the leaves of the network.

Example 1. A phylogenetic network for a given binary matrix M is illustrated in Figure 1.



M	c_1	c_2	c_3	c_4	c_5	c_6
s_1	1	0	0	0	0	0
s_2	0	1	0	0	0	0
s_3	0	0	1	0	0	0
s_4	0	1	0	0	0	1
s_5	0	1	0	1	0	0
s_6	0	1	0	1	0	1
s_7	0	0	1	0	0	1
s_8	0	0	1	0	1	1

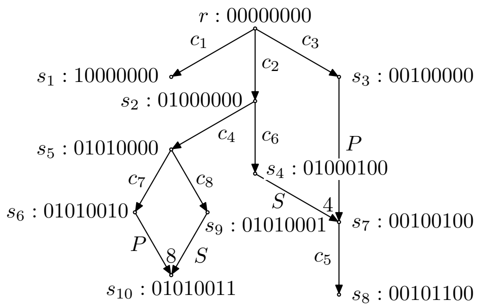
Figure 1: A phylogenetic network for matrix M. In the network, each mutation edge is labeled by a character c_i, $i \in \{1, \ldots, 6\}$; recombination edges are labeled by P and S respectively; the integer label above each recombination vertex represents the recombination point. Each species labels either a leaf or an internal vertex of the network.

Definition 4. (Galled-tree network) In a phylogenetic network N, let v be a vertex that has two paths out of it that meet at a recombination vertex x (v is the least common ancestor of the parents of x). The two paths together form a *recombination cycle* C. The vertex v is called the *coalescent vertex*. We say that C contains a character i, if i labels one of the edges of C.

A phylogenetic network is called a *galled-tree network* if no two recombination cycles share an edge. A recombination cycle of a galled-tree network is sometimes referred to as a *gall*.

Note that in the original definition of galled-tree network (cf. [25]) it is required that recombination cycles do not share vertices. It is easy to see that our modification is only a minor difference (one can be transformed to the other easily) introduced for technical reasons.

Example 2. A GTN for a binary matrix M is illustrated in Figure 2. Note that the two recombination cycles in this phylogenetic network do not share any edges. On the other hand, the phylogenetic network in Figure 1 is not a GTN as the two recombination cycles share one edge.



M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
s_1	1	0	0	0	0	0	0	0
s_2	0	1	0	0	0	0	0	0
s_3	0	0	1	0	0	0	0	0
s_4	0	1	0	0	0	1	0	0
s_5	0	1	0	1	0	0	0	0
s_6	0	1	0	1	0	0	1	0
s_7	0	0	1	0	0	1	0	0
s_8	0	0	1	0	1	1	0	0
s_9	0	1	0	1	0	0	0	1
s_{10}	0	1	0	1	0	0	1	1

Figure 2: A galled-tree network for matrix M. Two galls labeled by the set of vertices $\{s_5, s_6, s_9, s_{10}\}$ and $\{r, s_2, s_4, s_3, s_7\}$ do not share any edges in the network.

Definition 5 (Explains)**.** Given a haplotype matrix B, we say that a phylogenetic (galled-tree) network N with m characters *explains* B if each row of B is a label of some vertex in N.

Given a genotype matrix A, we say that A can be *explained* by phylogenetic tree/galled-tree network if there exist a haplotype matrix B inferred from A such that B can be explained by a phylogenetic tree/galled-tree network.

No we are ready to state the haplotyping problem via (simple) galled-tree networks.

Problem 1. (Galled-tree network haplotype (GTNH) problem) Given a genotype matrix A, decide if A can be explained by a galled-tree network.

A haplotype matrix B explainable by an SGTN inferred from a genotype matrix A will be called a *solution.*

In [6], we showed that this problem is NP-complete even in the case when each gall contains at most 4 mutations, and our preliminary results show that it is so also in the case of at most 3 mutations per gall. In this paper, we will consider the last open case, haplotyping via galled-tree networks with galls containing exactly two mutation edges (note that galls with only one mutation edge can easily eliminated, hence would not appear in the tree with the minimum number of galls). We will call such galls *simple galls* and the galled-tree networks with simple galls only, the *simple galled-tree networks.* The problem of deciding whether a genotype matrix can be explained by a simple GTN will be called the *SGTNH problem.* We will show that the SGTNH problem is polynomial for matrices with the weak property defined later.

2.3 Characterization

In [4], we have developed a characterization of the existence of galled-tree network for a given set of haplotypes. We will use this characterization to infer haplotypes from genotypes which can be explained by a galled-tree network. To state the characterization we will need the following two definitions.

Definition 6 (Column-restricted submatrix)**.** Let S be a subset of characters. The matrix $A[S]$ is the sub-matrix of A restricted to the columns in S. By $M[S] - x$, we denote the sub-matrix of $M[S]$ from which we remove all rows whose strings are identical to x.

Definition 7 (Conflict graph)**.** Given a haplotype (genotype) matrix M, we say that the characters c_1 and c_2 *conflict* in M if $M[c_1, c_2]$ contains (induces) all three pairs $[0, 1]$, $[1, 0]$ and $[1, 1]$. The *conflict graph* G_M of M has the vertex set correspond to the character set and for every two characters c_1 and c_2, (c_1, c_2) is an (undirected) edge of G_M if they conflict.

Note that in perfect phylogeny no two characters can conflict (four-gamete test).

Now we are ready to state the characterization.

Theorem 2. *([4]) Given a haplotype matrix M, M can be explained by a galled-tree network if and only if every nontrivial component (having at least two vertices) K of the conflict graph G_M satisfies the following conditions:*

*(1) K is bipartite with partitions L and R such that all characters in L are smaller than all characters in R (*the ordered component property*); and*

(2) there exists a row $x \neq 0^{|K|}$ such that $M[K] - x$ has no conflicting characters.

For simple galled-tree networks this characterization yields much simpler claim.

Corollary 1. *Given a haplotype matrix B, B can be explained by a simple galled-tree network if and only if every non-trivial component of the conflict graph of B has two vertices (B contains only isolated edges).*

2.4 Condition on the input genotype matrix

We will restrict the problem on genotype matrices which satisfy the following condition.

Definition 8 (Weak property)**.** Given a genotype matrix A, we say that A has the *weak property* if every pair of characters containing $[2,2]$ induces either both $[0,1]$ and $[1,0]$, or $[1,1]$. For any two columns c_i, c_j, let *indicator* ind_{ij} be 1 if c_i, c_j induce $[0,1]$ and $[1,0]$, and 0, otherwise. Note that in the second case, they have to induce $[1,1]$.

We will call a genotype matrix satisfying the weak property *simple*. Let us state two observations about the weak property.

- First, a simple genotype matrix A has the following useful feature: if B and B' are two haplotype matrices inferred from A such that $I_B(x_{rc_1c_2}) \neq I_{B'}(x_{rc_1c_2})$, for some $x_{rc_1c_2} \in \mathcal{X}_A$, then c_1 and c_2 conflict either in B or in B'.
- Second, note that the weak property is equivalent to the following condition: "every column containing 2 must also contain 1" (under assumption that A does not contain any rows with only one 2 which are easy to eliminate from the matrix without affecting the solutions to the problem).

Hence, this property is not very restricting. Indeed, the data set presented in [3] and most of the simulation data sets which we randomly generated genotypes from Hudson's simulation program [16] satisfy the weak property. The data in [3] contains a set of genotypes for 103 SNPs which are partitioned into 11 blocks. The experiments show that genotype matrices of each block, and also of the whole data have the weak property. The Hudson's simulation program [16] generates random haplotype matrices explainable by perfect phylogenies. We randomly choose vertices of their phylogenies and replace them with random galls with two mutations. The genotype data is then generated by combining rows of thus obtained haplotype matrices (each haplotype is repeated 3 or 4 times). Our experiments show that all 60 generated genotype matrices have the weak property (sizes of matrices: 300×20, 300×30 and 300×40; 20 matrices per size).

3 The algorithm

Our algorithm has three phases. In the first phase, it will produce a genotype matrix A' from A (by replacing some rows with 2's with rows inferred from them) such that A' can be explained by a SGTN if and only if A does, and each row of A' has either zero, three or four 2's. In the second phase, we will further replace more rows with 2's so that a new genotype matrix A'' satisfies a special condition (SC) (described later). In the last phase, the algorithm will infer B from A'' (if possible) using a reduction to a hypergraph covering problem. Finally, a solution for A'' can be easily transformed to a solution for A.

3.1 Phase 1: Eliminating rows with less than three or more than four 2's.

The following two claims show how to transform A to A' eliminating all rows with one or two 2's.

Claim 1. *Given an $n \times m$ genotype matrix A. Assume that A has a row r which contains one 2. Let A' be the matrix obtained from A by replacing r with the $2 \times m$ matrix inferred from r. Then A can be explained by a galled-tree network N if and only if A' can be explained by N.*

Proof. First, note that $\mathcal{X}_A = \mathcal{X}_{A'}$. Hence, there is a one-to-one correspondence between matrices B and B' inferred from A and A': $I_B = I_{B'}$. The matrix B' can be obtained from B by duplicating two rows which obviously does not affect whether the matrix can be explained by a galled tree network. □

Claim 2. *Given an $n \times m$ genotype matrix A with the weak property. Assume that A has a row r which contains exactly two 2's (in columns c_1 and c_2). Let A' be a matrix obtained from A by replacing r with the $2 \times m$ matrix R inferred from r such that $I_R(x_{rc_1c_2}) = ind_{ij}$. Then A can be explained by a simple galled-tree network if and only if A' can be explained by a simple galled-tree network.*

Proof. First, assume that A' can be explained by a simple galled-tree network N, i.e., there is a haplotype matrix B' inferred from A' which can be explained by N. Consider a matrix B inferred from A such that $I_B = I_{B'} \cup \{(x_{rc_1c_2}, ind_{rc_1c_2})\}$. Obviously, B' can be obtained from B by duplicating the two rows $2r-1$ and $2r$ (inferred from row r).Obviously, B can be explained by N.

For the converse, assume that A can be explained by a simple galled-tree network N, i.e., there is a haplotype matrix B inferred from A which can be explained by N. By Corollary 1, G_B contains only isolated edges. It is easy to see that the conflict graphs G_B and $G_{B'}$ differ by at most one edge, in particular $G_{B'}$ might not contain the edge (c_1, c_2). Hence, $G_{B'}$ contains only isolated edges and by Corollary 1, B' can be explained by a simple galled-tree network. □

Next, we will deal with the rows with five or more 2's. We will say that two solutions B and B' *agree* on set of characters C in row r, if for every $c, c' \in C$, $I_B(x_{r_{cc'}}) = I_{B'}(x_{r_{cc'}})$.

Claim 3. *Given a genotype matrix A with the weak property. Let r be a row of A with at least five 2's, say in columns $C = \{c_1, \dots, c_5\}$. Then either A cannot be explained by a simple galled-tree network, or every two solutions B and B' agree on C in r, i.e., there is unique way resolving these five 2's in a row r.*

Proof. Assume to the contrary that there are two matrices B, B' inferred from A explainable by an SGTN and two columns $c_i, c_j \in C$ such that $I_B(x_{rc_ic_j}) \neq I_{B'}(x_{rc_ic_j})$. By Corollary 1, characters c_i, c_j conflict in at least one of the matrices B and B', say in B. Let k_1, k_2, k_3 be the remaining three characters in C. Since, B can be explained by an SGTN, pairs c_i, c_{k_t} and c_j, c_{k_t}, $t = 1, 2, 3$, cannot conflict in B. By Lemma 1, for every $t = 1, 2, 3$, either $I_B(x_{rc_ic_{k_t}}) \neq I_{B'}(x_{rc_ic_{k_t}})$, or $I_B(x_{rc_jc_{k_t}}) \neq I_{B'}(x_{rc_jc_{k_t}})$. Consequently, by the weak property, either c_i, c_{k_t} or c_j, c_{k_t} conflict in B'. Hence, there are at least 3 conflicts in B' among five characters in C, a contradiction. □

Using the above claims we have the following procedure converting matrix A to matrix A' with desired properties.

Procedure 1. **Elimination of less than three or more than four 2's in rows.**

1. *Replace every row r with one 2 (two 2's) with the $2 \times m$ matrix R inferred from r (such that $I_R(x_{rc_1c_2}) = ind_{ij}$).*
2. *For every row r with 2's in columns $c_1, c_2, \dots, c_k$, where $k \geq 5$:*
   (a) *For every 5-tuple $c_i, \dots, c_{i+4}$, where $i = 1, \dots, k-4$, replace row r in $A[c_i, \dots, c_{i+4}]$ with all (16) possible inferrings of $r[c_i, \dots, c_{i+4}]$ to obtain 16 matrices $A_1^i, \dots, A_{16}^i$.*

(b) *Find the matrix $A^i_{j_i}$ whose conflict graph has only isolated edges. By Claim 3, at most one of our matrices has this property. If there is no such matrix, the original inferring problem has no solution: report* fail. *Otherwise, the matrix $A^i_{j_i}$ determines unique values of $I_B(x_{rc_pc_q})$ in every solution B, for every $p, q \in \{i, \ldots, i+4\}$.*

(c) *If the values of $I_B(x_{rc_pc_q})$ for $p, q \in \{1, \ldots, k\}$ are determined consistently over all 5-tuples considered, replace r by two rows inferred from r according to these values. Otherwise, there is no solution: report* fail.

The correctness of Step 1. is guaranteed by Claims 1 and 2. The correctness of Step 2. can be formally proved using Claim 3 and arguments similar to those used in proofs of Claims 1 and 2.

Observe that the conflict graphs of A and A' are isomorphic.

3.2 Phase 2: Eliminating some rows with three or four 2's.

We may assume that matrix A has zero, three or four 2's in every row. In this phase, we will resolve rows with three or four 2's, if there is a unique way of doing that. For this we use the following two claims.

Claim 4. *Given a simple genotype matrix A with the weak property. Let r be a row with three 2's, say in columns $C = \{c_1, c_2, c_3\}$. Then*

(1) if the conflict graph of $A[C]$ has one edge then there is a unique solution B for $A[C]$;

(2) if the conflict graph of $A[C]$ has no edge and the sum of indicators of C, $ind_{12} \oplus ind_{23} \oplus ind_{13} = 0$ then there is a solution B for $A[C]$ which has no conflicts, and every other solution B' for A agrees with B on C in r, i.e., there is a unique way how to resolve these three 2's in the row r in A;

(3) if the conflict graph of $A[C]$ has no edge and the sum of indicators of C, $ind_{12} \oplus ind_{23} \oplus ind_{13} = 1$ then there are three solutions B_1, B_2, B_3 for $A[C]$ such that c_1, c_2 conflict only in B_1 (and not in B_2 and B_3), c_2, c_3 only in B_2, and c_1, c_3 only in B_3, and every solution B for A agrees with one of them on C in r.

Proof. If the conflict graph of $A[C]$ contains two edges then A is not simple. If it contains one edge then all solutions for A must agree on C in r, case (1) of the claim. Assume now that the conflict graph of $A[C]$ has no edges.

Let B be a matrix inferred from A. Then if $ind_{ij} \oplus I_B(x_{rc_ic_j}) = 1$, c_i and c_j conflict in B. Observe that

$$(ind_{12} \oplus I_B(x_{rc_1c_2})) \oplus (ind_{23} \oplus I_B(x_{rc_2c_3})) \oplus (ind_{13} \oplus I_B(x_{rc_1c_3})) = ind_{12} \oplus ind_{23} \oplus ind_{13}\,.$$

First, assume that $ind_{12} \oplus ind_{23} \oplus ind_{13} = 0$. If B is a solution then $ind_{12} \oplus I_B(x_{rc_1c_2}) = ind_{23} \oplus I_B(x_{rc_2c_3}) = ind_{13} \oplus I_B(x_{rc_1c_3}) = 0$, i.e., all solutions (if they exist) must agree on C in r. In addition, it is easy to construct a particular solution B for $A[C]$ satisfying the requirements of case (2) of the claim as follows. Set $I_B(x_{rc_1c_2}) = ind_{12}$, $I_B(x_{rc_2c_3}) = ind_{23}$ and $I_B(x_{rc_1c_3}) = ind_{13}$, resolve every other row containing three 2's in $A[C]$ in the same way, and then resolve every row containing two 2's in $A[C]$ such that it will not introduce a conflict (similarly as was done in Claim 2).

Second, assume that $ind_{12} \oplus ind_{23} \oplus ind_{13} = 1$. If B is a solution then exactly one of $ind_{12} \oplus I_B(x_{rc_1c_2})$, $ind_{23} \oplus I_B(x_{rc_2c_3})$ and $ind_{13} \oplus I_B(x_{rc_1c_3})$ is equal to 1, i.e., exactly one pair in C is in conflict in both $B[C]$ and B. In addition, it is easy to construct particular solutions B_1, B_2, B_3 for $A[C]$ satisfying the requirements of case (3) of the claim as in the previous case. □

Claim 5. *Given a genotype matrix A with the weak property. Let r be a row with four 2's, say in columns $C = \{c_1, c_2, c_3, c_4\}$. Then*

(1) if the conflict graph of $A[C]$ contains at least one isolated edge then there is a unique way how to resolve these four 2's in the row r;

(2) if the conflict graph of $A[C]$ contains no edge then there are at least two triples in C with the sums of indicators equal to 0 or all four triples have the sums of indicators equal to 1. In the former case there is a unique way how to resolve these four 2's in the row r. In the later case there are three solutions B_1, B_2, B_3 for $A[C]$: pairs c_1, c_2 and c_3, c_4 conflict only in B_1 (and not in B_2 and B_3), pairs c_2, c_3 and c_1, c_4 only in B_2, and pairs c_1, c_3 and c_2, c_4 only in B_3, and every solution B for A agrees with one of them on C in r.

Proof. If the conflict graph of $A[C]$ contains two adjacent edges then the matrix A is not simple. Second, assume that it contains at least one isolated edge, say c_1, c_2. Then, by Claim 4, triples of 2's in c_1, c_2, c_3 and c_1, c_2, c_4 in row r have unique ways how to be resolved in all solutions for A. Therefore, all four 2's in r have unique way how to be resolved, case (1).

Hence, assume that the conflict graph of $A[C]$ does not contain any edge. To prove the first part of case (2) it is enough to observe that there cannot be exactly one triple in C with the sum of indicators equal to 0. If there are at least two such triples then by Claim 4, these two triples in row r have unique ways how to be resolved, and the claims follows. Finally, if all four sums of indicators are equal to 1 then by Claim 4, each triple has three solutions in r and there are exactly three ways (described in the statement of the claim) how to combine solutions from different triples to a solution for C in r. $\square$

Using the above two claims we have

The following procedure converts matrix A' to matrix A'' with zero, three or four 2's in every row such that **(SC)**: *for every triple of 2's in one row, say in columns c_1, c_2, c_3, the sum of indicators of c_1, c_2, c_3 is 1 and the conflict graph of A'' has no edges between c_1, c_2, c_3.*

Procedure 2. **Elimination of triples of 2's not satisfying (SC).**

1. *For every row with four 2's, say in columns $C = \{c_1, c_2, c_3, c_4\}$, such that at least one triple from C has the sum of indicators 0 or at least one pair from C conflicts in A', replace r by two rows inferred from r which do not violate any condition. This can be done in the unique way described in Claim 5.*

2. *For every row with three 2's, say in columns $C = \{c_1, c_2, c_3\}$, such that the sum of indicators of C equals to 0 or at least one pair from C conflict in A', replace r by two rows inferred from r which do not violate any condition. Again, this inferring is unique as described in Claim 4.*

Observe that the conflict graphs of A' and A'' are again isomorphic.

3.3 Phase 3: Reducing to a hypergraph covering problem

To complete the algorithm, we will characterize conflict graphs of all haplotype matrices inferred from the genotype matrix A'' in terms of a hypergraph coverings. Then we will use this characterization to reduce the SGTNH problem to a hypergraph covering problem. We will also provide a polynomial algorithm for the hypergraph covering problem. Let us start with the definition of a genotype hypergraph.

Definition 9 (Genotype hypergraph)**.** Given an $n \times m$ genotype matrix A with the weak property, the *genotype hypergraph* H_A of A has the set of characters $\{1, \dots, m\}$ as a vertex set, and for every row r of A containing 2's, say in columns $c_1, \dots, c_k$ ($k = 3, 4$), there is a hyperedge $e_r = \{c_1, \dots, c_k\}$. Furthermore, for every two columns c and c' inducing $[0, 1]$, $[1, 0]$ and $[1, 1]$ in A (conflicting in A), there is a hyperedge $\{c, c'\}$ in H_A. The hypergraph H_A does not contain any other hyperedges. A hyperedge with k vertices will be also called a k-edge. Note that H_A has $O(n)$ k-edges, $k = 3, 4$ and $O(m)$ 2-edges. We say that a graph G on the vertex set $V(H_A)$ *covers* a hypergraph H_A with hyperedges of cardinality $2, 3, 4$, if G can be obtained as follows:

- for every 2-edge $\{c_1, c_2\}$ of H_A, add the edge (c_1, c_2) to G;
- for every 3-edge $\{c_1, c_2, c_3\}$ of H_A, add exactly one of the edges (c_1, c_2), (c_2, c_3) and (c_1, c_3) to G;
- for every 4-edge $\{c_1, c_2, c_3, c_4\}$ of H_A, add exactly two disjoint edges (d_1, d_2) and (d_3, d_4) such that $\{d_1, d_2, d_3, d_4\} = \{c_1, c_2, c_3, c_4\}$ to G.

Note that any graph covering the hypergraph H_A contains all edges of the conflict graph G_A. Consider the following hypergraph covering problem.

Problem 2 (Hypergraph Covering (HC) Problem)**.** Given a hypergraph H with $2, 3, 4$-edges, determine whether there is a graph G that covers H and all its components have at most 2 vertices.

As a consequence of Claims 4 and 5 we have the following characterization of solutions to the SGTNH problem.

Lemma 3. *Let A be a simple genotype matrix, which satisfies the weak property, the (SC) property, and has zero, three, or four 2's in each row. A haplotype matrix B is a solution to the SGTNH problem for A if and only if G_B is a solution to the HC problem for genotype hypergraph H_A.*

Proof. First, assume B is a solution to the SGTNH problem for A (can be inferred from A via an SGTN). By Corollary 1 every non-trivial component of the conflict graph G_B has 2 vertices, i.e., is a single edge. By Claim 4(3) and Claim 5 (2) the conflict graph G_B is a solution to the HC problem for H_A.

For converse, let G be a graph inferred from H_A such that all its components have at most two vertices. By the same claims as in the paragraph above, there is a matrix B with the conflict graph G which can be inferred from A via an GTN. Since all components of G have at most two vertices, the GTN has only simple galls. □

Now, we are ready to reduce GTNH problem, in the case when the input genotype matrix has the strong diagonal property, to a graph theoretical problem. Later, we will show that this problem can be solved in polynomial time.

Lemma 4. *If the input genotype matrix has the weak property then the GTNH problem can be solved using the GI problem.*

Proof. Consider an input genotype matrix A with the strong diagonal property. First, if there is a row with more than four 2's, by Corollary 3, A cannot be explained by a galled-tree network. Second, replace each row that has one or two 2's with the corresponding inferred rows using Claims 1 and 2, respectively. Let the new matrix be A'. Note that A' can be explained by a galled-tree network if and only if A does. Each row of A' has either no, three or four 2's, hence the genotype hypergraph $H_{A'}$ contains only hyperedges of cardinality between 2 and 4, and all 2-edges are forced. We will show that A' can be explained by a galled-tree network if and only if it is possible to infer a graph G with all components of order at most 2 from $H_{A'}$.

First, assume that A' can be explained by a galled-tree network, i.e., there is a haplotype matrix B inferred from A' which can be explained. By Lemma 3, the conflict graph G_B can be inferred from $H_{A'}$. By Corollary 1, all components of G_B have cardinality at most 2.

For converse, let G be a graph inferred from $H_{A'}$ such that all its components have at most two vertices. By Lemma 3, there is a haplotype matrix B inferred from A' such that G is its conflict graph. By Corollary 1, B can be explained by a galled-tree network. $\square$

3.4 Solving Hypergraph Covering Problem

Now, it suffices to show that the HC problem can be solved in polynomial time. We will need the following definitions describing different configurations in a genotype hypergraph.

Definition 10. Let H be a hypergraph. A *3-edge-path* of length k is a subhypergraph of H consisting of k 3-edges $e_1 = \{v_1, v_2, v_3\}$, $e_2 = \{v_3, v_4, v_5\}$, $\ldots$, $e_k = \{v_{2k-1}, v_{2k}, v_{2k+1}\}$, where vertices $v_1, \ldots, v_{2k+1}$ are all distinct. In the case when all vertices $v_1, \ldots, v_{2k+1}$ are distinct except $v_1 = v_{2k+1}$, we call this subhypergraph a *3-edge-cycle*, cf. Figure 3.

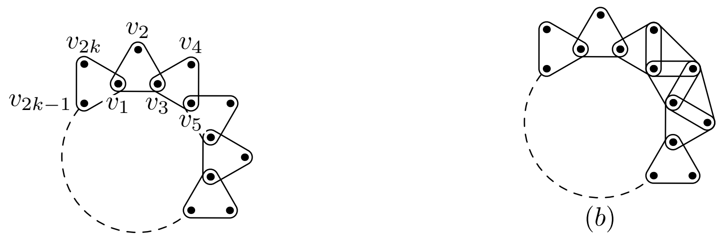


Figure 3: (a) Example of 3-edge-cycle. (b) 3-edge-cycle with a 3-edge-chain on it.

Observation 1. *If the input hypergraph for the HC problem is a 3-edge-cycle $e_1 = \{v_1, v_2, v_3\}$, $e_2 = \{v_3, v_4, v_5\}$, $\ldots$, $e_k = \{v_{2k-1}, v_{2k}, v_1\}$, where $k \geq 3$ then there are exactly two solutions. One solution consists of edges $\{(v_i, v_{i+1});\ i \text{ is odd}\}$ and the other of edges $\{(v_i, v_{i+1});\ i \text{ is even}\}$, where $v_{2k+1} = v_1$. In any solution, all vertices are incident to an edge of the solution.*

The following lemma describes the easiest case which can be solved by first identifying and covering cycles, and then greedily remaining 3-edges.

Lemma 5. *If the hypergraph H contains only 3-edges and any two of them are either disjoint or share exactly one vertex, then the HC problem for H can be solved in time $O(n^2)$.*

Proof. We give a polynomial algorithm for inferring a graph G with components of order at most 2 from the input hypergraph H. It is easy to see that we may assume that H is connected, otherwise we would consider each component separately. Initially, let G be the empty graph on vertices of H. If H does not have any 3-edge-cycle, pick a 3-edge $e = \{u, v, w\}$ which has at most one vertex incident to other 3-edges of H ("a leaf"). If there is no such vertex then H contains only 3-edge e, and any pair of vertices of e can be added to G to form a solution. Otherwise, let u be the vertex of e incident to other 3-edges. Remove e together with vertices v and w from H and add the edge (v, w) to G. We repeat the process above until all 3-edges of H are removed. Since, in each iteration, after adding an edge to G, the corresponding end vertices are removed from H, all edges in G are independent. Hence, G forms a solution.

Next, suppose H contains a 3-edge-cycle C. We will need the following simple observation.

Observation 2. *Let G be a graph on vertices of H. If a 3-edge $e = \{u, v, w\}$ of H has exactly one of its vertices, say u, incident to an edge of G, then any solution of H containing G must contain (v, w).*

By Observation 1, there are only two graphs which can be inferred from C. Hence, one of them is a part of G. In both of them, every vertex of C is incident to an edge. Based on this fact, we will show that for any other 3-edge, there is at most one choice for an edge of the 3-edge to be added to G. In case there is no choice for a 3-edge then no graph satisfying the requirements can be inferred from H.

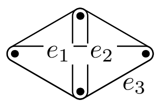
First, pick one of the solutions for C and add it to G. Remove all 3-edges of C from H. If there is a 3-edge $e = \{u, v, w\}$ in H such that exactly one of its vertices, say u, is incident to an edge of G then remove e from H and add (v, w) to G. Repeat this until there is no such 3-edge in H. If there is no 3-edge left in H, we are done. Note that by the construction there are no incident edges in G, hence G is a solution to the HC problem on H.

Now, assume there is a 3-edge left in H. By applying Observation 2 in each iteration of the above process, it is easy to see that all edges in G (with exception of the edges from C) are necessarily in the solution of the HC problem for H. If there is a 3-edge in H with at least two vertices incident to edges of G then no edge of this 3-edge can be added to G, hence, there is no solution to the HC problem on H. Hence, we can assume that all remaining 3-edges in H have no vertex incident to edges of G. Since H was initially connected, there is an 3-edge e in H incident to 3-edge $e' = \{u, v, w\}$ already removed from H. Let u be the common vertex. Since e has no vertex incident to edge of G, the chosen edge from e' is (v, w). However, that implies that u was incident to some edge in G at the moment when e' was processed. This is a contradiction, since u must be still incident to the same edge of G. $\square$

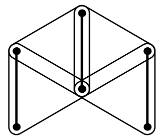
We say that two 3-edges are *doubly incident* if they share two vertices. A set S of 3-edges is *doubly connected* if for any pair e and e' in S, there exists a sequence of consecutively doubly incident 3-edges from S starting with e and ending with e'. A *doubly connected component* is a maximal doubly connected set. Note that the remaining 3-edges in H can be uniquely partitioned into doubly connected components. We have the following observation.

Observation 3. *Let H be a hypergraph with only 3-edges and C any doubly connected component of H. Covering of any 3-edge of C inductively forces coverings of remaining 3-edges of C which will either lead to a contradiction or unique solution for C.*

Three pairwise doubly incident 3-edges are called a *doubly incident 3-edge-cycle*, cf. Figure 4(a). Figure 4(b) shows a hypergraph with a unique solution (depicted with bold 2-edges) which plays an important role in the following lemma which characterizes solutions for doubly connected components.



(a)



(b)

Figure 4: (a) Doubly incident 3-edge-cycle; (b) Forcing configuration of four doubly connected 3-edges.

Definition 11. Let H be a hypergraph. A *3-edge-chain* of length k is a subhypergraph of H composed of k 3-edges $e_1 = \{v_1, v_2, v_3\}$, $e_2 = \{v_2, v_3, v_4\}$, $\dots$, $e_k = \{v_k, v_{k+1}, v_{k+2}\}$, where vertices $v_1, \dots, v_{k+2}$ are all distinct, cf. Figure 5. The vertices v_1 and v_{k+2} are the *end vertices* of the chain.

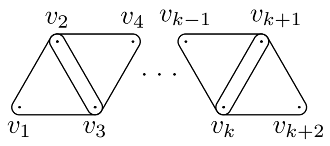


Figure 5: Example of a 3-edge-chain of length k.

We have the following observation about solutions to the HC problem on 3-edge-chains.

Observation 4. *If the input hypergraph for the HC problem is a 3-edge-chain $e_1 = \{v_1, v_2, v_3\}$, $e_2 = \{v_2, v_3, v_4\}$, $\dots$, $e_k = \{v_k, v_{k+1}, v_{k+2}\}$, where $k \geq 3$ then there are exactly two solutions. One solution consists of edges $\{(v_i, v_{i+1});\ i \text{ is odd}\}$ and the other of edges $\{(v_i, v_{i+1});\ i \text{ is even}\}$. If k is odd then all but one of the end vertices are incident to an edge in both solutions. If k is even, in the first solution all vertices are incident to an edge, while in the second solution all but two end vertices are incident to an edge.*

Lemma 6. *Let H be a hypergraph with only 3-edges without doubly incident 3-edge-cycles such that every pair of vertices belongs to at most two 3-edges. If C contains the hypergraph depicted in Figure 4(b) then there is either no way or a unique way how to cover C. Otherwise, C is a 3-edge-chain.*

Proof. First, if C has at most two 3-edges, it is a 3-edge-chain, cf. Figure 6.

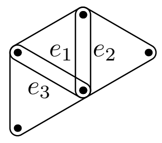


Figure 6: A 3-edge-chain.

Second, consider a doubly connected set S with three 3-edges e_1, e_2, e_3. We may assume that e_1, e_2 (respectively, e_2, e_3) are doubly incident. Since, by assumption, e_1, e_3 are not doubly incident, we can assume that all doubly connected sets with three 3-edges form a 3-edge-chain, cf. Figure 7.

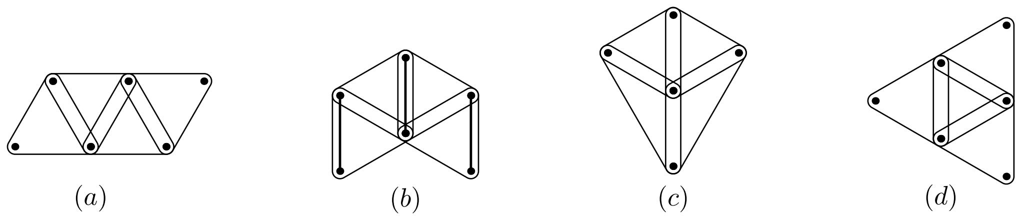


Figure 7: All possible cases for a doubly connected component with four 3-edges.

Now, consider a doubly connected set S with four 3-edges. There are only four possibilities how this set can look like, depicted in Figure 7. In the first case, we have a 3-edge chain. In the second case, by Observation 3, there is either no covering or a unique covering of C. In the third and fourth cases there is no solution. Hence, we can assume that all doubly connected sets with four 3-edges form a 3-edge-chain. It follows that every doubly connected component is a 3-edge chain. □

Now, we are ready to attack the HC problem in the general case.

Theorem 7. *The HC problem can be solved in $O(n(n+m))$ time.*

Proof. We give a polynomial algorithm for constructing a covering graph G with components of order at most 2 from an input hypergraph H. Initially, let G be the empty graph on vertices of H. First, we remove all 2-edges from H and place them to G. The algorithm will deal with the hyperedges of H one by one using a set of rules. Each processed hyperedge is removed from H and either

(T1) some pairs of vertices contained in the hyperedge are placed as edges to G so that the conditions of Definition 9 are satisfied, or

(T2) the choice for pairs of vertices from this hyperedge is postponed and binded to decision on the choice for another hyperedge still in H, or

(T3) an auxiliary hyperedge is added to H and the choice for pairs from the original hyperedge is binded to decision on the choice for this new hyperedge.

The first happens only if there is a unique choice for covering the processed hyperedge. Hence, in the moment when G contains two incident edges, it follows that there is no solution to the HC problem on H. It will be obvious from the algorithm that after applying all possible rules (in given order) either all hyperedges are processed, or all unprocessed edges are 3-edges, any two unprocessed hyperedges share at most one vertex and no unprocessed hyperedge and edge of G share a vertex. Hence, we can then apply the algorithm of Lemma 5 on the modified H (containing only unprocessed edges). Union of G and a solution from the algorithm of Lemma 5 gives a solution to the HC problem.

The set of rules:

1. If there is an hyperedge e contained in another hyperedge f, we proceed as in the case (T2). In particular, we remove e and bind the choice for pairs in e to the choice for pairs in f, i.e., covered pairs in e are exactly the pairs covered in f which are contained in e. From now on, we can assume that no hyperedge is contained in another.

2. If there is a pair of vertices u, v contained in at least three 3-edges, then remove all of them from H and add the edge (u, v) to G, cf. Figure 8.

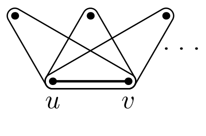


Figure 8: The pair u, w contained in at least three 3-edges.

3. If H contains doubly incident 3-edge-cycle, cf. Figure 4(a), remove 3-edges of the cycle from H and add a new 4-edge into H consisting of the four vertices of the cycle. It is easy to see that this does not influence the solution.

4. For each 4-edge that intersect with other hyperedges in H or edge in G, its covering is uniquely determined or there is no covering for it and the algorithm fails. Consult Figure 9 for all possible cases.

(a) If two 4-edges share exactly one vertex, then there is no solution to the HC problem.

(b) If two 4-edges share exactly two vertices, remove them from H and add the pair of shared vertices as an edge to G. Furthermore, add the two remaining pairs (one per 4-edge) as edges to G. This results in covering of the two 4-edges.

(c) If two 4-edges share exactly three vertices, then there is no solution.

(d) If a 4-edge shares exactly one vertex with a 3-edge $e = \{u, v, w\}$. Let u be the common vertex. Then remove e from H and add edge (v, w) to G.

(e) If a 4-edge shares exactly two vertices with a 3-edge, remove both of them from H and add the common pair together with the other disjoint pair of the 4-edge as edges to G.

(f) If a 4-edge shares exactly one vertex with a 2-edge in G, there is no solution.

After applying the above rules as many times as possible, the only 4-edges left in H are isolated from other hyperedges and also from edges in G. Hence, for each of them, we can arbitrarily pick covering pairs of edges and remove it from H. Thus, we can assume there are no 4-edges left in H.

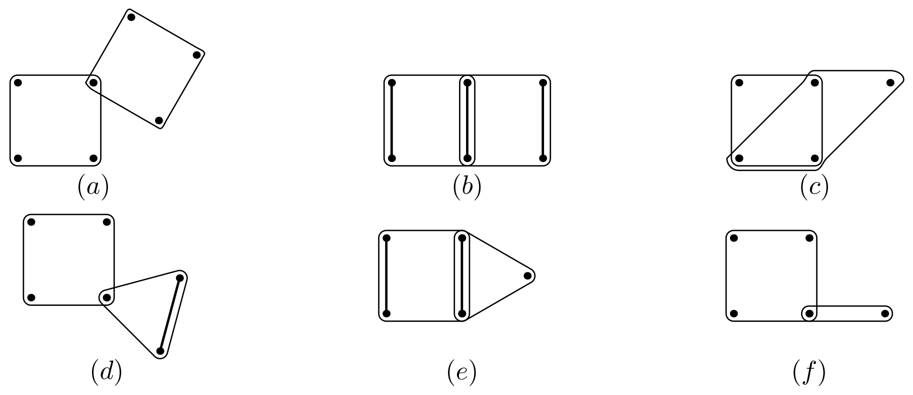


Figure 9: An 4-edge intersecting other hyperedges. The thick edges show the covering if it exists. In cases (a), (c) and (f) there is no solution.

5. If a 3-edge intersects a 2-edge in G in a single vertex, remove the 3-edge from H and add the edge formed by the remaining two vertices of the 3-edge to G. Apply this rule whenever a new edge is added to G.
6. Consider a doubly connected component C. By Lemma 6, either there is a unique or no way of covering C, or C is a 3-edge chain. In the former case, remove C from H and add 2-edges of the solution for C to G or return fail. Assume the second case. By Observation 4, if there is a 3-edge not in C containing an interval vertex of C, this 3-edge is uniquely covered. Hence, remove it from H and add the pair of its other two vertices as an edge to G. Now, all 3-edge chains can share only their end vertices.

If a component C is a 3-edge-chain of even length, by Observation 4, there is a solution s for C which does not contain the end vertices, cf. Figure 10(a). Covering C in this way will not affect the existence of the solution for H, since if there is a solution to H which contains the other solution for C then by replacing it with s results in a new solution which in addition has smaller number of edges. Hence, assume all doubly connected components are 3-edge-chains of odd length.

If there are two vertices which are end vertices of at least three 3-edge-chains, there is no solution. If there are two vertices which are end vertices of exactly two 3-edge-chains, then by Observation 4,

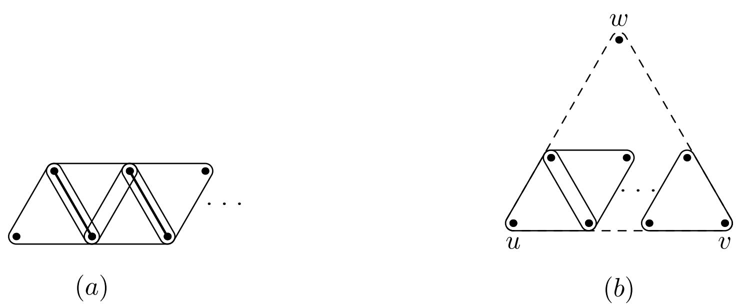


Figure 10: (a) An example of smaller solution to a 3-edge-chain of even length. (b) Replacement of a 3-edge-chain of odd length by a new 3-edge.

there are only two solution for their union and each of them contains the two vertices. Hence, we can arbitrarily pick one of the solutions and add it to G and remove both 3-edge-chains from H.

Finally, we proceed as in case (T3): replace every 3-edge-chain by a 3-edge having the two end vertices u, v of the 3-edge-chain and one new vertex w, cf. Figure 10(b). The solution for the 3-edge-chain is binded to the new 3-edge as follows: if the edge (u, v) is picked to cover the 3-edge, then cover the 3-edge-chain in any of the two possible ways; if the edge (u, w) is picked, cover the 3-edge-chain by the solution that contains u; and similarly for the edge (v, w).

After application of all rules above (which can be done in polynomial time), H will satisfy requirements of Lemma 5, and hence the solution can be found (if one exists) in polynomial time. Finally, the resolution of binded hyperedges can be done in polynomial time as well. To analyze the running time of the algorithm, let us look at individual steps. Steps 1.–3. runs in time $O(n^2)$, 4. and 6. in $O(n)$, and 5. in time $O(nm)$. The algorithm of Lemma 5 takes time $O(n^2)$ and finally, resolution of binded hyperedges can be done in $O(n)$ time. $\square$

Note that the solution constructed by the above algorithm has the minimum number of edges, hence the corresponding galled-tree network has the minimum number of recombinations.

To summarize the last phase, let us state the procedure inferring the genotype matrix A''.

Procedure 3. **Inferring matrix using genotype hypergraph:**

1. *Construct the genotype hypergraph for A'' (see Definition 9).*

2. *Use the algorithm described in Theorem 7 and Lemma 5 to find covering of the genotype hypergraph (with the minimal number of edges in the cover).*

3. *Using the covering determine how to infer every row with 2's.*

Complexity Analysis: The preprocessing which includes: determining values of indicators, listing of 2's in each row and computing the conflict graph of A, takes time $O(nm^2)$. Phase 1 takes time $O(nm)$, Phase 2 $O(n)$ and Phase 3 (solving HC problem) $O(n^2 + nm)$. Thus, the algorithm runs in time $O(n^2 + nm^2)$.

3.5 Experimental Results

We implemented the algorithm and performed the following experiments on simulation data. We compared our algorithm's performance with PHASE v2 [24, 23]. The results show that our algorithm seems promising for the corresponding haplotyping problem.

Table 1: Comparison of accuracy between our algorithm (SGTN) and PHASE on the five sets of matrices. The first row of the table lists the size of the matrices that were generated using Hudson program. The second row lists the average size of each group of matrices after applying our random algorithm and removed the duplated columns. Each rate is normalized by the number of genotype matrices in that set.

	20*20		30*20		30*40		100*50		100*100	
	30*14		45*16		75*20		199*33		199*49	
	SGTN	PHASE	SGTN	PHASE	SGTN	PHASE	SGTN	PHASE	SGTN	PHASE
error rate	0.005	0.008	0.012	0.013	0.002	0.000	0.007	0.002	0.002	0.001
discrepancy	0.09	0.13	0.250	0.283	0.070	0.000	0.545	0.149	0.198	0.126
switch accuracy	0.995	0.992	0.988	0.987	1	1	0.993	0.998	0.998	0.999
time	0.54s	14.2s	0.49s	26.8s	0.43s	28.57s	0.83s	204s	2.95s	368.36s

In particular, the data is prepared in three steps. First, binary matrices, each having a perfect phylogeny tree, are generated using Hudson's [16] simulation program with recombination rate being 0. Then, for each matrix's perfect phylogeny tree, we randomly replace nodes on it with simple galls by adding new columns to matrices with each newly added column introducing a conflict between itself and an existing column in the matrix (thus, adding recombinations). Last, for each haplotype in the matrix, we randomly repeat it for 2 to 6 times and pair haplotypes to generate genotype matrices.

We infer haplotypes from the generated genotype matrices that have weak property using our algorithm and compare our algorithm's performance with PHASE base on three standards [24]: the *error rate*: the proportion of genotypes having more than two 2's whose haplotypes are incorrectly inferred; the *discrepancy rate*: the sum of frequency differences for each individual haplotypes (simulated or true ones); and the *switch accuracy* ([17]) of the inferred haplotypes is defined as $(h - w - 1)/(h - 1)$, where h is the number of 2's in a genotype g and w is the number of wrongly inferred 2's for g. Using Hudson's simulation algorithm and applying our random procedure to generate genotypes, we generated five sets of 100 matrices. The experimental results are shown in Table 1.

The results show that our algorithm have comparable accuracy with PHASE while runs tens to hundreds times faster.

4 Conclusions

We considered the problem of haplotyping via galled-tree networks introduced in [11]. We have shown that this problem can be solved in polynomial time for the input genotype matrices with the weak property and the galled-tree networks with two mutations per gall. Together with the result in [6] and preliminary results, where we showed that the problem is NP-complete if the size of galls (the number of mutation edges) is either not restricted or restricted to any $k \geq 3$, we have almost completely determined the complexity of the problem. The remaining open problem is the complexity of the haplotyping via simple galled-tree networks for the genotype matrices without the weak property.

References

[1] V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph. Haplotyping as perfect phylogeny: A direct approach. *Journal of Computational Biology*, 10(3-4):323–340, 2003.

[2] P. Bonizzoni, G. D. Vedova, R. Dondi, and J. Li. The haplotyping problem: An overview of computational models and solutions. *Journal of Computer Science and Technology*, 18:675–688, 2003.

[3] M. Daly, J. Rioux, S. Schaffner, T. Hudson, and E. Lander. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29(2):229–232, 2001.

[4] A. Gupta, J. Maňuch, L. Stacho, and X. Zhao. Characterization of the existence of galled-tree networks. In *Proceedings of the 4th Asia-Pacific Bioinformatics Conference*, pages 297–306, 2006.

[5] A. Gupta, J. Maňuch, L. Stacho, and X. Zhao. Algorithm for haplotype inferring via galled-tree networks with simple galls (extended abstract). In S. Istrail, P. Pevzner, and M. Waterman, editors, *Proc. of International Symposium on Bioinformatics Research and Applications (ISBRA) 2007*, volume 4463, pages 121–132. Lecture Notes in Bioinformatics, 2007.

[6] A. Gupta, J. Maňuch, L. Stacho, and X. Zhao. Haplotype inferring via galled-tree networks is NP-complete. In *Proc. of COCOON 2008, LNCS 5092*, pages 287–298, 2008.

[7] A. Gupta, J. Maňuch, L. Stacho, and X. Zhao. Haplotype inferring via galled-tree networks using a hypergraph covering problem for special genotype matrices. *Discrete Applied Mathematics*, page (to appear), 2008.

[8] D. Gusfield. Haplotyping as perfect phylogeny: conceptual framework and efficient solutions. In *RECOMB '02: Proceedings of the sixth annual international conference on Computational biology*, pages 166–175, New York, NY, USA, 2002. ACM Press.

[9] D. Gusfield. An overview of combinatorial methods for haplotype inference. In *Computational Methods for SNPs and Haplotype Inference*, volume 2983 of *Lecture Notes in Computer Science*, pages 9–25. Springer Berlin / Verlag, 2004.

[10] D. Gusfield. Optimal, efficient reconstruction of root-unknown phylogenetic networks with constrained and structured recombination. *J. Comput. Syst. Sci.*, 70(3):381–398, 2005.

[11] D. Gusfield, S. Eddhu, and C. Langley. Optimal, efficient reconstruction of phylogenetic networks with constrained recombination. *Journal of Bioinformatics and Computational Biology*, 2(1):173–213, 2004.

[12] D. Gusfield and S. H. Orzack. *Handbook of Computational Molecular Biology*, chapter Haplotype inference, pages 18–1–18–28. CRC Computer and Information Science Series. Chapman & Hall, 2005.

[13] B. V. Halldórsson, V. Bafna, N. Edwards, R. Lipert, S. Yooseph, and S. Istrail. A survey of computational methods for determining haplotypes. In *Computational Methods for SNPs and Haplotype Inference*, volume 2983 of *Lecture Notes in Computer Science*, pages 26–47. Springer Berlin / Verlag, 2004.

[14] J. Hein. Reconstructing evolution of sequences subject to recombination using parsimony. *Mathematical Biosciences*, 98:185–200, 1990.

[15] J. Hein. A heuristic method to reconstruct the history of sequences subject to recombination. *Journal of Molecular Evolution*, 36:396–405, 1993.

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

[17] S. Lin, D. Cutler, M. Zwick, and A. Chakravarti. Haplotype inference in random population samples. *American Journal of Human Genetics*, 71:1129–1137, 2002.

[18] R. D. Mitra, V. L. Butty, J. Shendure, B. R. Williams, D. E. Housman, and G. M. Church. Digital genotyping and haplotyping with polymerase colonies. In *Proceedings of the Nationlal Academy of Sciences of the United States of America*, volume 100, pages 5926–5931, 2003.

[19] N. Patil, A. Berno, D. Hinds, W. Barrett, J. Doshi, C. Hacker, C. Kautzer, D. Lee, C. Marjoribanks, D. McDonough, B. Nguyen, M. Norris, J. Sheehan, N. Shen, D. Stern, R. Stokowski, D. Thomas, M. Trulson, K. Vyas, K. Frazer, S. Fodor, and D. Cox. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, 294(5547):1719–1723, 2001.

[20] Y. Song and J. Hein. On the minimum number of recombination events in the evolutionary history of DNA sequences. *Journal of Mathematical Biology*, 48:160–186, 2003.

[21] Y. S. Song, Y. Wu, and D. Gusfield. Algorithms for imperfect phylogeny haplotyping (IPPH) with a single homoplasy or recombination event. In *WABI*, pages 152–164, 2005.

[22] S. Sridhar, G. E. Blelloch, R. Ravi, and R. Schwartz. Optimal imperfect phylogeny reconstruction and haplotyping. In *Proc. of the Computational Systems Bioinformatics Conference (CSB06)*, pages 199–210, 2006.

[23] M. Stephens and P. Donnelly. A comparison of bayesian methods for haplotype reconstruction from population genotype data. *American Journal of Human Genetics*, 73:1162–9, 2003.

[24] M. Stephens, N. J. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction from population data. *Amer. J. of Human Genetics*, 68(4):978–989, 2001.

[25] L. Wang, K. Zhang, and L. Zhang. Perfect phylogenetic networks with recombination. *Journal of Computational Biology*, 8(1):69–78, 2001.