HAPLOTYPE INFERRING VIA GALLED-TREE NETWORKS

by

Xiaohong Zhao

B.Sc., Peking University, 2000

M.Sc., The University of British Columbia, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Computing Science

APPROVAL

Name:	Xiaohong Zhao
Degree:	Doctor of Philosophy
Title of thesis:	Haplotype Inferring via Galled-Tree Networks

Examining Committee: Dr. David Mitchell
Chair

Dr. Arvind Gupta, Senior Supervisor

Dr. Ladislav Stacho, Supervisor

Dr. Felix Breden, Supervisor

Dr. Nansheng(Jack) Chen, SFU Examiner

Dr. Eleazar Eskin, External Examiner,
Assistant Professor,
Computer Science and Human Genetics,
University of California, Los Angeles

Date Approved: April 17, 2008

Abstract

Despite the great progress biomedical research has made, the root causes of common human diseases remain largely unknown. It is widely believed that mutations in DNA sequences, especially those interrupting the composition of genes, can cause illness. Searches for DNA variants for some rare monogenic diseases have been successful. However, it is more complicated to determine the genetic basis of common "multi-factorial" diseases, which are the result of multiple genetic variants and environmental factors. In such cases, each individual causative variant may have modest effect on the phenotype and therefore is hard to detect. Haplotypes are believed to be promising genetic markers to tackle the problem. Understanding of haplotype structures will provide fundamental insights into the pathogenesis, diagnosis and treatment of common diseases.

Nevertheless, current in vitro techniques for direct haplotype sequencing are prohibitive due to their high cost and low throughput. So research has turned to inferring haplotypes computationally. In this thesis, we use the galled-tree network approach on the haplotyping problem – the computational problem of inferring haplotypes from genotypes assuming the evolutionary history for the original haplotypes satisfies the galled-tree network model. The model is an extension to the perfect phylogeny model, which was widely utilized in haplotype inferring. While galled-tree network (GTN) model is promising in its ability to handle broader range of biological data, it also increases the complexity of the corresponding haplotyping problem. It has been an open problem as to how hard this problem is, and whether there is efficient algorithm to solve it. In this thesis work, we characterized both the problem of galled-tree network's existence for binary matrices and the problem of utilizing GTN to infer haplotypes from genotypes. Furthermore, we developed a polynomial algorithm for a special case of the galled-tree network haplotyping problem. The evaluation results are promising.

Keywords:

haplotypes; human genetic variation; haplotype inferring; galled-tree networks; graph theory; complexity theory.

Acknowledgments

I owe deep debt of gratitude to my senior supervisor, Dr. Arvind Gupta, for his guidance not only on my Ph.D. research but also on my career development. His extensive support and insightful suggestions kept me being positive at times of confusion and difficulties throughout my Ph.D. years. I am deeply grateful to my supervisor, Dr. Ladislav Stacho, for his continuous support and inspiration. Working with him has given me the precious opportunity to see the beauty of graph theory and other areas of mathematics. I would also like to thank our postdoc fellow, Dr. Ján Maňuch, for the countless hours of discussions and generous help. I am sincerely thankful to my supervisor, Dr. Felix Breden, for his helpful insights, enthusiasm and constant support. This thesis would not be possible without any of them.

I am also thankful to Dr. Nansheng (Jack) Chen and Dr. Eleazar Eskin for their patience in reading my thesis, their intriguing questions, and their valuable suggestions which have served to improve this thesis.

My gratefulness also goes to Dr. David Mitchell and Dr. Evgenia Ternovska for their inspiration and help on my research. I would also like to thank Ms. Val Galat, Ms. Gerdi Snyder and Ms. Kersti Jegger for their assistance throughout my Ph.D. study. My appreciation also goes to my colleagues and friends at the COMBI Lab and the School of Computing Science at SFU.

Last but not least, I would like to thank my parents, for their never-failing believe in me and their unconditional support. I am especially grateful to my husband, Jian, for his always being there for me - rain or shine.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Despite the great progress biomedical research has made, the root causes of common human genetic diseases remain largely unknown. It is widely believed that mutations in DNA sequences, especially those interrupting the composition of genes, can cause illness. Searches for DNA variants for some rare monogenic diseases have been successful. However, it is more complicated to determine the genetic basis of common "multi-factorial" diseases, which are the result of multiple genetic variants and environmental factors. In such cases, each individual causative variant may have modest effect on the phenotype and therefore is hard to detect. Discovering these genetic factors will provide fundamental insights into the pathogenesis, diagnosis and treatment of common diseases (see [86] for some recent progress of studies on human genome variants and their roles in helping us understand complex diseases and different human traits).

Traditional disease association studies are aimed at finding correlations between individual genetic variants and a certain disease. This can entail the full genome sequencing of numerous individuals and is thus impractical at present. In fact, the target variants are often limited to some previous assumptions on where causative variants can be located. This strategy can easily miss true causative genetic variants for diseases. Researchers have gained substantial insight into the variations in human genomes through the tremendous progress of the Human Genome Project. This has enabled scientists to dramatically narrow down the search space for causative genetic variants and therefore make the approach promising.

One key discovery is that common variants account for 90% of all human genome variations. Based on the common variant/common disease assumption, studies on these common variants should provide knowledge of most of the genetic variants that are associated with particular diseases. Furthermore, most of the common variants originally arose from single mutation events, and are therefore associated with the adjacent variants that were present together on the ancestral chromosome region ([18]). In the case when some "real" genetic variant is not covered by the common variants in some association study, the above feature helps researchers to identify the causative region which contains the causative genetic variant. This makes the following "indirect" approach feasible in studying candidate genetic variants that cause diseases. Instead of searching one individual variant at a time, "indirect" association studies use a set of sequence variants in the genome to serve as genetic markers in order to detect association with the disease. This approach has already led to the discovery of common genetic factors for several common diseases ([18]). Determining the common patterns of DNA sequence variations of individuals on each of their chromosomes (the haplotypes) is one of the main goals in the next stage of the Human Genome Project and this body of work is now encompassed in the international HapMap project (see [57]). Most genes will contain at least one or more genetic markers because of their high density (one such marker is expected in about every 300 nucleotides)([14]). Therefore, the associations found with these markers make the subsequent work of verifying the functionality of related genes much easier. By making the information freely available to researchers around the world, the project provides invaluable resources that can guide the design and analysis of genetic association studies.

Until recently, haplotypes were poorly understood. Experimentation allows for cost-effective determination of genotype information (the combined information for an individual across both homologous chromosomes). However, in-vitro techniques for haplotyping are prohibitive due to cost and low throughput. There is strong interest in the development of algorithmic tools [79] to infer haplotypes from genotypes.

In this Chapter, we give basic genetic background of the haplotyping problem, and illustrate some molecular haplotyping methods. Next, we review computational haplotyping problems, and describe some main algorithms respectively. Finally we state some interesting open problems in this field.

1.1 Basic Background on Genetics and Genomics

We give basic background knowledge on genetics and genomics in this section. More detail can be found in [14, 25, 41, 18, 12, 63, 85, 110, 20, 100, 113].

1.1.1 DNA, Genes and Proteins

DNA (deoxyribonucleic acid), is a macromolecule that encodes all of our inherited information and is responsible for developing and directing the functioning of the human body. A DNA sequence contains millions of bases, where each base is one of the four nucleotides; denoted A, C, G and T. DNA molecules are mainly stored in the nucleus of the cell. They, together with the proteins that cover them, form the chromosomes. In humans, there are 46 chromosomes in total. Among them, 2 are sex chromosomes (X and Y) which determine the gender of an individual (XX for females, XY for males), and 44 are non-sex chromosomes (autosomes). The 46 chromosomes comprise two nearly identical copies of the whole genome: one copy is inherited from the father (via a set of 22 autosomal chromosomes and an X or Y chromosome) and the other copy from the mother (via another 22 autosomal chromosomes and an X chromosome). Each pair of autosomal chromosomes inherited from the father and mother are homologous: they look similar and comprise the same genes. This is also true for the two X chromosomes in females.

A gene is a stretch of DNA that contains the code for the production of a protein, which is a part of the machinery for a cell to function properly in its environment. In particular, a gene encodes all the information that determines the exact composition of a specific protein, as well as the regulatory instructions of when and in what quantity the protein will be produced.

Variations on genes, especially on the regulatory or coding regions of genes, can affect the production of proteins, which can eventually affect phenotype: the observable or measurable characteristics of humans. It is widely expected that the study of genetic variations will result in breakthroughs in understanding the pathogenesis of diseases. Indeed, some genetic variants are already found to be the cause for certain diseases. For instance, sickle cell anemia, which was the first genetic disease to be understood at the genomic level ([14]), is caused by a single nucleotide's variation that results in a one amino acid difference which leads to the malfunction of the protein – hemoglobin. Other examples include Type 1 diabetes ([25]) , Alzheimer's disease ([105]) and stroke ([40]) (see [18] for more examples). These causative genetic variants can in turn serve as targets for drug discovery. In addition, knowledge of the role genetic variants play in drug responses can help divide patients into several groups and help doctors choose feasible medical treatments based on their genetic variants.

1.1.2 From SNPs to Haplotypes

Association studies are often applied in the search for candidate genetic variants for common diseases. The technique is to find an association between genetic variants and a disease, by comparing a group of affected individuals with a group of unaffected controls. Undertaking association studies through testing each individual candidate variant could entail substantial expense as it requires search of the entire genome for numerous individuals in both groups. At present, this search is restricted in the candidate regions that are suggested by previous genetic hypotheses. Furthermore, this approach is not feasible for multi-factorial diseases, where each genetic variant has very modest effect on the disease and thus is difficult to detect.

Most human geneticists believe that for a majority of diseases, there are relatively common genetic variants that are causative in disease susceptibility – this is known as the common variant/common disease hypothesis ([12]). A systematic survey for all the common variants of the population will dramatically reduce the workload for researchers who are searching for causative genetic variants for diseases.

There are several forms of genetic variations. When the variation involves just one nucleotide, it is the Single Nucleotide Polymorphism (SNP). SNPs are the most common genetic variation, comprising about 90% of all differences in the population ([18]). SNPs often have two values (*alleles*) in the whole population, where each allele appears in >1% of the population. The most common allele is known as the *major allele*, and the other *minor allele*. Since humans are diploid organisms (there are 2 copies for each homologous chromosome), for a chosen SNP site, a person can have one of several possible combinations of alleles (*genotypes*) in a pair of chromosomes: both alleles are major or minor (*homozygous*) or one is major and one is minor (*heterozygous*).

The International SNP Map Working Group ([90]) estimates that 93% of genes contain a SNP, and 98% are within 5 kilobases of a SNP. Nearly every human gene or genomic region is marked by a SNP ([15]). Not all SNPs are found in every population, but about 82% of SNP variants appear in more than 10% of the global human population. The distribution of other SNPs in individual populations is not known ([15]). On average, a SNP occurs in every 300 bases. Using SNPs as genetic markers provides a map of common variations with very dense resolution. Therefore with a high likelihood, the method can be successful in detecting causative variants (genes) for complex diseases. SNPs can be characterized by whether they are coding or non-coding (i.e. whether they are located in genes or not). Some coding SNPs can alter protein sequences and therefore invoke protein malfunctions (e.g. [14]). Some SNPs in non-coding regions can also cause complex genetic diseases, e.g. SNPs that sit in gene regulating regions. These SNPs can alter the quantity of expressed genes and thereby trigger diseases (e.g. [98]). While SNPs have been used to successfully identify genes associated with diseases, for multi-factorial diseases, individual SNPs may have poor predictive power in disease association studies ([20, 100, 63]). In particular, this method does not provide clear end points as to which genes are causative: true associations might be missed by the incomplete information provided by individual SNPs; negative results do not rule out association involving other nearby SNPs and positive results could just point out some variants that are inherited together with the true causative variants but not the actual causative variants themselves ([20]).

A set of SNP alleles on a chromosome is a ***haplotype***. Recent studies ([60, 85, 20]) show that many chromosomes are composed of "blocks" of nucleotides, with each block having just a few (usually around 4 to 5 ([20])) common haplotypes among all the population. This is consistent with the hypothesis that most common variants arose from some single historical mutation event, and are inherited together with the nearby variants that were present on the ancestral chromosome region when the mutation happened. SNPs on each haplotype block have strong association with each other, which makes haplotypes very important markers in detecting disease associated genes ([69, 100, 20, 38, 29, 100, 63, 68]). Indeed, association studies on haplotypes provides crisp regions for candidate causative genetic regions which are essential for subsequent tests. Studying haplotype structures is also essential in understanding the evolutionary history for the population. Due to the limited diversity for each haplotype block, a few SNPs (tag SNPs) are usually sufficient to distinguish different haplotypes. It is estimated that between 50,000 and 100,000 SNPs should be sufficient to represent enough genetic diversity to support "genome scans" for association between haplotypes and disease susceptibility ([37]). Compared with the approximately 10 million SNPs across all human genomes, the number of tag SNPs is much smaller, and therefore will dramatically reduce the search space for disease association studies. Construction of a haplotype map, which records the common patterns of DNA sequence variation in the human genome, is the main aim of the International HapMap Project ([57]).

Though haplotypes are promising in serving as efficient genetic markers, their structures are still poorly understood. In-vitro techniques for sequencing genotypes (the combination of alleles for a pair of homologous chromosomes) have become more prevalent over the years. However, direct haplotyping methods (that is, molecular haplotyping methods) are still not feasible. Below, we review some well-known molecular haplotyping methods together with their limitations.

1.1.3 Molecular Haplotyping Methods

For diploid organisms, directly sequencing genotypes leads to ambiguities of the underlying haplotypes whenever there are more than one heterozygous site on the pair of chromosomes ([17]). There are quite a few molecular haplotyping methods developed to date (see [81, 13, 19] for brief reviews of molecular haplotyping methods) that try to overcome this difficulty. Here, we review the following three methods: cloning method, somatic cell hybrid method and allele-specific PCR method, which covers most of the basic ideas of molecular haplotyping methods.

The cloning method ([113]) usually has four steps: DNA sequences are first cut into shorter sequences by enzymes called *restriction endonucleases.* These enzymes recognize a certain sequence pattern in DNA molecules and make cuts in both strands at those sequences. The DNA fragments are then ligated to certain vector DNAs, that are capable of replicating in *E. coli.* In the 3rd step, the recombinant DNAs are replicated in bacteria, i.e., *E. coli.* Finally, the recombinant DNAs are screened and sequenced. The vector DNAs have resistance to several antibiotic genes. However, the ability to resist certain antibiotic genes is destroyed depending on where the foreign DNA (DNA fragments to be detected) sequences are ligated onto the vector DNA. In this way, different DNA clones are distinguished after they are transformed to bacteria for replicating. The cloning method can clone upto 50kb of DNA ([113]) based on the vector DNA's capacity of accepting foreign DNAs fragments. However, this size of DNA segments is relatively small and cannot meet the requirement for haplotyping.

Somatic cell hybrid technology ([41]) converts a diploid cell into haploid cells ([104, 26]) by constructing somatic cell hybrids, which often retain only a subset of human chromosomes. The hybrids are usually constructed by fusing human and mouse cells via a specific virus called the *Sendai virus.* Each Sendai virus has several points of attachment and it can simultaneously attach to two different cells when they are close. The Sendai virus is also small in comparison with a cell. Therefore, when a mouse cell and a human cell are attached by a Sendai virus, they become one. During the following cell divisions, for some unknown reasons, human chromosomes are gradually eliminated from the hybrid at random. With certain biochemical strategy, each hybrid will reach a stable state during the cell growing, with the entire set of mouse chromosomes and a small set of human chromosomes which differ for individual hybrids. Since there is recognizable difference between human and mouse chromosomes, human chromosome can be identified in each hybrid cell. Different hybrid cells are grown separately, and in total, they contain all the human chromosomes. Nevertheless, the somatic cell hybrid technique is very laborious and expensive ([19]).

Another molecular haplotyping technique is allele-specific PCR ([89, 78, 107]). Before we give the details about the technique of AS-PCR, there is a need to understand how PCR works. Polymerase Chain Reaction (PCR) is a molecular biology technique that replicates specific regions of DNA sequences ([113]). The technique works in cycles. In each cycle, there are three steps. First, target DNA sequences are heated to separate them into single strands. Then, the temperature is lowered to allow primers to anneal to the target DNA strands and initiate DNA synthesis (primers are short artificial DNA sequences that are complementary to the beginning or the end of the DNA fragment to be amplified). Lastly the temperature is raised a little to allow DNA synthesis which is catalyzed by a special DNA polymerase, the *taq polymerase*. Since alleles in each pair of chromosomes are highly similar, without any specificity, the PCR procedure will amplify both alleles of the pair of chromosomes and the investigator will only know that both alleles are present in an individual sample. In AS-PCR technique, allele-specific primers are designed to amplify only alleles in one of the chromosomes and thus obtain the corresponding haplotype sequences. However, there are several limitations to the method. One is that the Taq polymerase does not have an error checking facility as it adds nucleotides to the copies of DNA fragments. Therefore an error occurring in an early PCR cycle can result in large incorrect portions in the final product. Another difficulty is that it is difficult to amplify long stretch of DNA segments. A recent paper presented a technique that can deal with 10kb-long DNA segments. These limitations make the current AS-PCR technique not suitable for long-range haplotype sequencing.

There are other molecular haplotyping methods as well. However, none is widely used today. As with the techniques described above, these methods are usually hard to automate, low in throughput, cannot handle long range DNAs, and very expensive ([81, 68, 13]). On the other hand, computational methods are cost-effective and fast and many researchers have turned to such methods for haplotype inference.

In the following sections, we review computational haplotyping methods.

1.2 Overview of Computational Haplotyping Methods

Haplotyping methods mainly deal with two kinds of data: family genotype data (genotypes of individuals from a family) and population genotype data (genotypes of unrelated individuals). In general, family data is hard to collect and is not much more informative than population data ([81, 102]). Our focus of this review is on population based computational haplotyping problems.

Abstractly, the computational haplotype inferrence (HI) problem or haplotyping problem is described as follows: the input is genotype data collected from m sites (SNPs) of n individuals, where each individual is represented by a m-long vector. Each position of the vector can have one of three values that represents a combination of the two values on the pair of the chromosomes. In particular, 0 and 1 represent a *homozygous* site with minor or major allele on both chromosomes respectively and 2 represents a *heterozygous* site – both alleles are present at the site on the pair of chromosomes. For each individual, we would like to describe the alleles of the m sites on each copy of the two chromosomes separately, i.e., the haplotype. Indeed, the output of the problem is $2k$, $k \leq n$, m-long vectors, where each pair of vectors represent the two haplotypes that resolve (explain) the genotype of one individual. Obviously, genotype with h ***heterozygous*** sites would be consistent with any of the 2^{h-1} haplotype pairs that enumerated all the possibilities of $0, 1$s for the heterozygous sites. Note that solutions are invariant to swapping the two haplotypes of a genotype.

With different genetic models or hypothesis, reasonable solutions are defined accordingly. In the next several sections, we illustrate those widely used models/hypothesis and the corresponding algorithms. Note that there are many statistical methods developed for the haplotyping problem as well, among them are EM algorithm ([33, 108]) and Bayesian methods ([103, 102, 82, 74]). Some statistical software such as PHASE ([103, 102]), is widely used in biological applications. Nevertheless, these programs usually run in thousands of iterations in order to get a good solution. They are time-consuming and have relatively low throughput. In this Chapter, we focus on combinatorial algorithms developed for inferring haplotypes.

1.3 Parsimony Methods

In a seminal paper [17], Clark proposed a heuristic approach for inferring haplotypes, which is known as *inference rule* or *Clark's rule*. For the input genotypes set G, the algorithm first identifies all genotypes that have zero or one 2 in them. Obviously, haplotypes that resolve these genotypes are uniquely determined. These haplotypes form an initial set H of known haplotypes that are supported by the data. For each known haplotype h in H, the algorithm searches all the remaining unresolved genotypes in G, to see if there exists any genotype g such that g can be resolved by h and h', here h' is a new haplotype that could be added to H, when such an h' is to be one of the known haplotypes and g is removed from G. The procedure continues, until all the genotypes are resolved, or no new haplotype can be found. Those unresolved genotypes are called ***orphans***. Clark also showed a parsimony rule that was verified by simulation data and some real data sets: the solution that has fewest orphans after applying a sequence of Clark's rule generates the fewest incorrect haplotype assignments.

Gusfield ([47]) formalized the parsimony problem as that of finding a haplotyping solution with fewest orphans, and named it the *maximum resolution problem* (MR). He showed that the MR problem as well as several variations are NP-hard. These variations include the *unique expression MR problem* (UEMR), where every genotype in G can be explained by at most one pair of haplotypes in H; the *unique MR problem* (UMR), where every identical genotype in G must be explained by the same pair of haplotypes in H; and finally the *maximum resolution on G problem* (MRG), where the MR problem is converted to a graph problem as follows. Create a directed graph G_d containing a set of nodes N for each genotype in G and a set of nodes I for the completely determined haplotypes in H. For each genotype g in G that has h heterozygous sites, $R(g)$ is the set of the 2^h vertices, each labeled by the corresponding distinct, resolved vectors (haplotypes) created by setting the ambiguous positions in g to 0 or 1 in all possible ways. There will be an edge (v, v') if and only if v' is in some set $R(g)$ for a genotype g, v' is not in I and the pair v and v' represented vectors (haplotypes) explain g. The MRG problem is to find the largest number of vertices in G that can be reached by a set of vertex-disjoint, directed trees, where each tree is rooted at a node in I and where, for every genotype g in G, at most one node in $R(g)$ is reached. The worst-case reduction of the problem is in exponential time. Even then, without considering the complexity of the reduction step, the remaining optimization problem is also shown to be NP-hard. Gusfield gave an integer linear programming method to solve the problem and claimed this worked quickly and correctly in simulation data.

There are several concerns with the MR and its related parsimony models as summarized by Clark ([17]):

- The initial set H could be empty and the algorithm cannot even be started.
- There may be unresolved genotypes left.
- Haplotypes might be erroneously inferred if a crossover product of two actual haplotypes is identical to another true haplotype. Note that this did happen in the experimental studies, when the solution has the fewest unresolved genotypes.

Another parsimony HI problem that is widely mentioned in the literature is the *Pure Parsimony Problem*: find a solution to the HI problem that minimizes the total number of distinct haplotypes used. Unfortunately, it has also been shown to be NP-hard again by Hubbell ([49]). Several practical algorithms for the pure parsimony problem have been developed, among them are Gusfield ([49])'s integer linear programming formulation, and Wang et al. ([111])'s branch and bound method. Both techniques are claimed to be practical for data with real biology interests. Gusfield ([49]) pointed out that the speed of their algorithm improves with increasing recombination of the input data while the accuracy decreases. Wang ([111]) compared their algorithm with the statistical software package PHASE ([101, 103, 102]), and showed that their method has higher accuracy when the recombination is increased. Though both authors have noticed that recombination rate of the input data is related to the algorithms' performance, no explicit genetic model is used in their inference methods.

1.4 Perfect Phylogeny Haplotyping (PPH) Problem

At a meeting sponsored by NIH ([63]) held in 2001, researchers came to the conclusion that it is time to create a new map of the genome, one that describes its blocky structure ([60]). The new project is the international HapMap project ([57]), which represents the next high-priority phase of the Human Genome Project. The blocky structure refers to the property that genomic DNA can be partitioned into blocks, where nucleotides in each block are highly associated with each other and have few recombinations ([20, 100, 38, 60]). The key observation [48] is that without recombination, the backward history of a single haplotype with m SNP values from an individual is a path and the histories for two such individual haplotypes are two paths which merge at the most recent common ancestor of the two. In addition, the history satisfies the *infinite sites assumption.* That is, the m SNP sites are so sparse relative to the mutation rate, that in the time frame of interest at most one mutation can occur at any site. In other words, no back mutation happened in the evolutionary history for each individual haplotype. Gusfield ([48]) explored the algorithmic implications of this key observation: the evolutionary history for nucleotides in each block satisfies the *perfect phylogeny model* and utilized it in the haplotype inference problem. In the following section, we briefly review the perfect phylogeny problem and some related results.

1.4.1 The Perfect Phylogeny Problem

A phylogeny [34] is a tree representation of the evolutionary history for a set of taxa S. S can be a set of organisms, biological sequences, populations or languages. The phylogeny construction problem is among the basic computational problems both in biology and linguistics. Here, we assume each leaf vertex is labeled by a taxon from S, and each internal vertex including the root of a phylogeny is labeled by a taxon that represents hypothetical ancestor of the elements of S.

There are two main branches of phylogeny construction problems: one is *distance based* and the other is *character based.* Distance based construction problems have a distance matrix as input which describes some kind of distance between any two taxa in the problem (e.g., the alignment score between them or the fraction of residues that are different). The goal is to construct a phylogeny tree such that each leaf of the tree is labeled by a taxon, each edge of the tree is labeled by the distance between the taxa labels of its end points such that the distance between any two taxa on the tree is consistent with the input distance matrix. In the character based construction problem, the input is a character state matrix containing the state value for each character of each taxon. The output is a phylogeny tree that has each of its leaf labeled by a taxon, each edge labled by a character state representing the gain of that character state and the path from the root to each taxon contains exactly the character states gained during evolution. Here, we are concerned with the character based methods, since this is the one that is used intensively in haplotyping problems. We use taxa and species interchangeably in this thesis.

There are two kinds of characters: *cladistic characters* and *qualitative characters.* For a qualitative character, we have no information about the relationship between each state, while cladistic characters have such information illustrated in a *character state tree* for each character. The vertices on the tree are labeled by each state of the character. Cladistic characters can be further divided into *directed* and *undirected* cladistic characters based on whether or not each character state tree is rooted. Our focus is on the qualitative character based phylogeny construction problem.

Different criteria has been utilized to judge which tree is better for phylogeny construction problems. One of them, the *most parsimonious tree* , requires that the constructed tree has the minimum number of character state changes among all possible trees. Let C denote the set of characters that define the set of species S, and A_c the set of states for character c. Let $|C| = m$ and $max_c|A_c| = r$. Assume that $A_c \in \{1, 2, \dots, r\}$ and each species is identified with a vector $A_1 * \cdots * A_m$. Let $c(s)$ represent the state of character c for species s. Observe that for a phylogeny T for S, for every character c, the number of edges (u, v), where $c(s_u) \neq c(s_v)$, is at least $r_c - 1$ (assume that each state of c is exhibited by some species). If this is true for every character c in T, then T is the most parsimonious tree and T is called a *perfect phylogeny tree.* A nice survey of perfect phylogeny problem can be found at [34]. We give the definition of the problem below.

Definition 1. [2] The perfect phylogeny problem is to determine whether a given set of n distinct species S has a tree T with the following properties:

(C1) $S \subseteq V(T) \subseteq A_1 * A_2 * \dots A_m$,

(C2) Every leaf in T is in S, and

(C3) For every $c \in C$ and every $j \in A_c$, the set of all $u \in V(T)$ such that $c(u) = j$ induces a subtree of T.

If S has perfect phylogeny, the corresponding set of characters are *compatible.* The perfect phylogeny problem, was shown to be NP-complete by Bodlaender et al.[10] and independently, by Steel [97]. Polynomial algorithms are developed for problems with restricted number of states or restricted number of characters. Gusfield [46] designed an algorithm that can test if the given set of binary characters are compatible in time $O(nm)$, which is linear to the size of the input character state matrix. We give the definition of the problem below and describe the algorithm after that.

Problem 1 ([46]). Given an $n * m$ binary matrix M, determine whether there is a perfect phylogeny for M, and if so, build one.

In [46], the algorithm first preprocesses the input matrix M by sorting its columns. Considering each column of M as a binary number (with the most significant bit in row 1), sort these numbers into descending order with the largest number in column 1. For each set of duplicated columns, keep one of them and remove the others. The transformed matrix is called M'. The algorithm then verifies that the matrix M has a perfect phylogeny as follows. Let O be the set of cells in M' that have value 1. Let $L(i,j)$ be the largest index $k < j$ such that $M'(i,k) \in O$; set $L(i,j)$ be 0 if there is no such index k. For each column j, set $L(j)$ equal to the largest $L(i,j)$ such that $(i,j) \in O$. If $L(i,j) = L(j)$ for each cell $(i,j) \in O$, then there is perfect phylogeny for M, otherwise not. Let O_i be the set of objects with a 1 in column i. The checking procedure is supported by the following lemma.

Lemma 1. *M has a perfect phylogeny if and only if for every pair of columns i, j, either O_i and O_j are disjoint or one contains the other.*

If there is a perfect phylogeny for M, the algorithm constructs it as follows [46]:

(1) Create a node n_j for every column j of M'. For each node n_j such that $L(j) > 0$, add a direct edge $(n_{L(j)}, n_j)$, and label the edge with character j and all the indexes of all columns identical to column j (which were deleted in preprocessing). Create a root node r, and for each node n_j such that $L(j) = 0$, add a direct edge (r, n_j), and label the edge with j, and the indexes of all columns identical to j.

(2) For each row i, let c_i be the largest index such that cell $(i, c_i) \in O$ and let e be the edge labeled with character c_i. If the head of edge e is a leaf, then attach species i to that leaf. If the head of e is not a leaf, then create a new edge e' directed from the head of e and attach i to the new leaf created. Note that e' does not have a label on it. The resulting phylogenetic tree is a phylogenetic tree for matrix M.

Dress and Steel gave an $O(nm^2)$ algorithm for testing the compatibility of ternary characters [28]. Later, Kannan and Warnow gave an $O(n^2m)$ algorithm for quaternary characters [70]. Agarwala and Fernández-Baca [2] showed that for any fixed r the problem is polynomially solvable in time $O(2^{3r}(nm^3 + m^4))$. The bound was later improved to $O(2^{2r}nm^2)$ by Kannan and Warnow [71]. When the number of characters is fixed, a polynomial algorithm was found for the perfect phylogeny problems as well. The best bound, so far, of $O((r - n/m)^m rnm)$ was found by Agarwala and Fernández-Baca in 1996 ([3]).

For real biological data, perfect phylogeny is often too restricted and does not necessarily exist for a set of taxa. Various attempts have been made to find a polynomial algorithm for near-perfect phylogenies. In an attempt to minimize the amount of homoplasy (back mutation), Fernández-Baca and Lagergren defined the imperfection q of a phylogeny T for S as $length(T) - \sum_{c \in C}(r_c - 1)$, where $length(T)$ denotes the total number of character state changes among all edges of T. Note that this is the number of edges of T when each edge represents exactly one character state change. They showed [35] that the problem of determining whether a set of taxa admits a phylogeny with penalty q is polynomially solvable when q and the number of states for characters are fixed.

1.4.2 The PPH Problem

Preliminaries

Perfect phylogeny was intensively used in the haplotyping problem as the evolutionary model for haplotypes that form genotypes. We give the formal definition of the PPH problem below.

Definition 2. Given a genotype $n \times m$ matrix A with values in $\{0, 1, 2\}$, find a $2n \times m$ haplotype matrix B with values in $\{0, 1\}$, where rows $2i - 1$ and $2i$ of B represent haplotypes for the genotype in row i of A. We say that B is *inferred* from A if and only if for every character $c \in \{1, \ldots, m\}$,

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

Definition 3. Given a genotype matrix A, we say that A can be *explained* by a perfect phylogenetic tree if there exists a haplotype matrix B inferred from A such that B can be explained by a perfect phylogenetic tree.

Problem 2. (Perfect Phylogeny Haplotyping (PPH) problem) Given a genotype matrix A, decide if A can be explained by a perfect phylogenetic tree.

Combinatorial Algorithms for PPH Problem

The first such deterministic algorithm for PPH was presented by Gusfield ([48]). The algorithm reduces the PPH problem to a graph realization problem. The reduction is based on three main observations as quoted below [48]. Let S be the input genotype matrix, and for each genotype (row) g_i in S, let i and i' represent the corresponding haplotypes that form i.

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

(2) For any row g_i in S, and any column c, if $S(g_i, c) = 2$, then the path between i and i' must contain the edge labeled c.

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

Gusfiled showed that with the first observation, we can deduce the edges on the path from the root to the least common ancestor of i and i'. The order of these edges can be deduced based on the three observations as well. Furthermore, the paths defined this way form an "initial" unique perfect phylogeny that appears in every perfect phylogeny for S. In fact, Gusfield proved that if each column c in S contains at least one 1, then there is a unique perfect phylogeny for S and it can be found efficiently. The problem now is to determine the positions for the edges that are labeled by columns whose non-zero entries are just 2's. With these restrictions on the paths for the potential perfect phylogeny tree, the algorithm reduced the haplotyping problem to the graph realization problem. Let E_r be a set of r distinct integers. A "path-set" is an unordered subset P of E_r. A path-set is "realized" in an undirected, edge-labeled tree T consisting of r edges, if each edge of T is labeled by a distinct integer from E_r, and there is a contiguous path in T whose labels consist only of the integers in P.

Definition 4. (The Graph Realization Problem) Given E_r and a family $\Pi = P_1, P_2, \ldots, P_k$ of path-sets, find a tree T in which each path-set is realized, or determine that no such tree exists. Further, determine if there is only one such T, and if there is more than one, characterize the relationship between their trees.

The algorithm then applies known mathematical results on the graph realization problem ([9]) to find realizing trees for Π and therefore find the PPH solution for M. The complexity for the algorithm is $O(nm\alpha(nm, n))$, where α is the inverse Ackerman function, and hence this time bound is almost linear in the input matrix's size nm. The algorithm uses deep results in graph theory and is not easy to implement. Gusfield's paper was followed by an explosion of papers [8, 16, 30] studying the PPH problem where simpler algorithms running in $O(nm^2)$ time were presented. The recent paper [24] affirmatively answers the question of whether the PPH problem can be solved in linear time $O(nm)$. We briefly describe them in the following paragraphs.

Both Eskin et al.'s ([30]) and Bafna et al.'s ([8]) algorithms run in time $O(nm^2)$ and were claimed to be easier to implement than Gusfield's algorithm. Both algorithms heavily rely on the following theorem.

Theorem 2. *[46] A binary matrix B has perfect phylogeny if and only if for any two columns i and j, there are no four rows that have all the four value pairs [0,0], [0,1], [1,0] and [1,1] in i and j.*

The test for this feature is known as the "four-gamete test" in the biological literature. The columns i, j that contain all the four value pairs are *conflict* columns. The goal of both algorithms is to infer haplotypes without generating conflict columns. We assume the root to be the all-0 sequence in the following illustrations. As an example, we describe Bafna et al.'s algorithm below.

First, we introduce some notation that will be used later in the illustration of the algorithms.

Definition 5. Given an $n * m$ binary matrix A, let S be a subset of characters of A. The matrix $A[S]$ is the sub-matrix of A restricted to the columns in S.

Definition 6. Given 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 $[x, y]$, $[2, y]$ or $[x, 2]$.

Note that for two columns c_1, c_2, if they induce value pair [1,1] or both pairs [0,1] and [1,0], then the pattern of how to resolve $[2, 2]$ on c_1 and c_2 is forced in order to create a perfect phylogeny.

Bafna et al. [8] builds a genotype graph $G_A(C, E_f \cup E_n)$ from A, where the vertex set is the set of columns in A. A column pair $(c_1, c_2) \in E_f$ if and only if $A[\{c_1, c_2\}]$ contains the value pair [2, 2] and there is a forced pattern for them to be resolved. $(c_1, c_2) \in E_n$ if and only if $A[\{c_1, c_2\}]$ contains the value pair [2, 2] and there is no forced pattern for them to be resolved. Assume we label each edge in G_A as follows: let 0(1 respectively) mean that the two columns are to be resolved equally, i.e. 2 2 $\rightarrow \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$ (unequally, i.e., 2 2 $\rightarrow \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ respectively). Bafna showed that every edge in a component of E_f can be uniquely labeled with 0 or 1. Note that edges in a component of E_f are not necessarily from E_f. Shrink each component of E_f to be a new vertex, and let the new (multi)-graph be H. The algorithm then arbitrarily chooses a spanning tree T of H. Note that, for each set of parallel edges in H, there is only one of these in T. Each assignment of 0, 1 values to the edges of T determines a haplotype resolution to A. [8] also proved a non-trivial upper bound on the number of PPH solutions. They showed that if every column of M contains at least one 0 and one 1 entry, then there is at most one PPH solution for M. Furthermore, if M has k columns that each contains both a 0 and 1 (and a 2 by prior assumption), then M has at most 2^{m-k-1} PPH solutions. The upper bound is essentially $2^{|T|}$, and $(m - k - 1)$ is the upper bound for $|T|$, the number of edges in T.

One main step for both Eskin et al.'s and Bafna et al.'s algorithms is to find the relation between any two sites (columns) in the input genotype matrix. It is proved in [6] that algorithms like this are not likely to be fulfilled in linear time. [6] also showed that if there are multiple PPH solutions, then to find one that is the most parsimonious in terms of the number of the distinct haplotypes is still NP-hard.

Finally, Ding et al. [24] gave an $O(nm)$ algorithm to the PPH problem. The algorithm builds and uses a directed rooted graph: *shadow tree*. Ding et al. showed that every PPH solution is contained in the final shadow tree produced by the algorithm; every tree contained in the final shadow tree structure, obtained by performing some standard operations on the tree corresponds to a distinct PPH solution. They also gave experimental results showing its efficiency.

One problem with the above algorithms is that few of them were tested on real population data. In fact, real haplotypes often do not fit the perfect phylogeny model, due to the noise inherent in real data, i.e, the missing values and the way to seperate long range DNA data into blocks where the nucleotides on them are considered to have high association with each other and thus satisfy perfect phylogeny model. Several papers present algorithms that are some extensions to the perfect phylogeny model ([31, 56, 55]). In [31, 55], Eskin and Halperin allow a small number of haplotypes to violate the perfect phylogeny model. In particular, they define an error threshold e. For two columns c_1 and c_2 in the input matrix, they say that c_1 and c_2 induce value pair $[x, y]$ where $x, y \in \{0, 1\}$, if and only if there are more than e rows that induce $[x, y]$ in these two columns. They claim that the majority of haplotypes fit the model much better. Halperin and Karp [56] considered the case when the input data has missing values. They studied the problem of constructing a perfect phylogeny compatible with a given set of partial haplotypes (or determine that none exists); and the problem of constructing a perfect phylogeny compatible with a given set of partial genotypes (or determine that none exists). Though both problems are shown to be NP-hard ([56, 97]), they designed polynomial algorithms that work for data which was generated under a probabilistic process. These algorithms are still based on the perfect phylogeny model. Though they claim data does not fit the perfect phylogeny model, no explicit genetic model is used to tackle certain biological events other than single mutation for each nucleotide. In the following sections, we review some computational haplotyping methods which take into account of additional biological events, such as back mutations, recurrent mutations and recombinations.

1.5 Imperfect Phylogeny Haplotyping Problem

There were several attempts in using imperfect phylogeny to explain the original haplotypes that formed the current genotypes. Some of them extends the perfect phylogeny model by allowing exactly one homoplasy event (a recurrent or back mutation for a certain character) and a recent study generalized the imperfectness to any constant q. We describe the two algorithms below.

Song et al. ([94]) designed an algorithm for haplotyping which allows exactly one homoplasy. The algorithm works as follows: it partitions the input genotype matrix M column-wise into two sub-matrices M_s and M_r. M_s has one single column of M and the rest denoted by M_r. The algorithm keeps partitioning the input genotype matrix this way, until it finds one partition such that M_r has a PPH solution (there exists a perfect phylogeny T that can explain the evolutionary history for the original haplotypes for M_r), and the column in M_s can be explained by adding a homoplasy event in T. Sridhar et al. ([96]) tackled the problem of using q-near-perfect phylogeny to do haplotyping, where q is the number of "extra" mutations needed to explain the data. The idea of their algorithm is similar with Song et al.'s. The algorithm traverses all the possibilities of the characters that mutate more than once and removes them from the input matrix. It then finds the PPH solution for the remainder and finally adds back the removed characters and performs haplotyping on the full matrix.

1.6 Galled-Tree Network Haplotyping (GTNH) Problem

When *recombination* occurs, the ancestral material for the present species comes from two instead of one parental species as in single mutations. Here, our focus is in single crossover events between two parental sequences (homologous chromosomes). The evolutionary history among such set of species can no longer be modeled by a rooted tree. Instead, a recombination topology takes into place. A *phylogenetic network* is a generalization of a phylogeny tree, allowing non tree-like structure properties that capture certain biological events such as recombinations, horizontal gene transfers and gene conversion. Hein ([58, 59, 93]) is among the first to give heuristic methods for constructing *phylogenetic networks* that derives a given set of sequences, minimizing the number of recombinations used. To date, no efficient algorithm is known for the problem. In particular, for binary sequences, Wang [112] proved it is NP-hard. There are several variations to the general phylogenetic networks, which are special constrained networks. Wang [112] defined a *perfect phylogenetic network*, where each character in the network will change at most once even after recombination, and recombination cycles are node-disjoint to each other. A poly-time algorithm to construct such network was given in [112] as well. Gusfield [54] extended this to *galled-tree network*, where the restriction for each character to only change once is removed and a poly-time algorithm to construct such a network for binary data is given. We will give the formal definition of the galled-tree network later.

Galled-tree network is believed to be realistic in modeling evolutionary history for species that undergoes a modest number of recombinations, (e.g. the evolutionary history for the human populations ([54])). There are other motivations for studying galled-tree networks, an important one among them is its nice combinatorial features. Gusfield [54] showed that if an input matrix M can be derived on a galled-tree, then it can be derived on a true tree with no cycles on it, and at most one back mutation per character. Furthermore, when M can be derived on a galled-tree, the galled-tree can be constructed efficiently (with poly-time algorithm) and the galled-tree uses the minimum number of recombinations over any phylogenetic network for M, even the ones that allow multiple-crossover events at each recombination node. This provides the only non-trivial case where phylogenetic networks with minimum number of recombinations can be efficiently constructed.

As illustrated in the above section , DNA data is blocky ([60]), with nucleotides on each block having very few recombinations and varieties. Furthermore, haplotype data on each block often does not fit the perfect phylogeny model ([31, 56, 55]). In fact, the International HapMap Consortium presented their findings in [64] on haplotypes after phase 1 of the HapMap project. They showed that the haplotypes can contain variant number of recombinations, though this number is usually small.

There is a greater need to understand the combinatorial properties of phylogenetic networks, especially the ones suitable for sequences with a modest number of recombinations. Using galled-tree network as the underlying model to infer haplotypes from genotypes has been an open problem for several years. However, no one was able to show either NP-hardness or give efficient algorithm to solve the problem. In Chapter 2 we give definitions and review research work that is related to galled-tree networks.

1.7 Main Results

The galled-tree network model can be applied to a wider range of data than the perfect phylogeny model, as when there is no recombination it reduces to perfect phylogeny. Furthermore, its feature that allows a small number of recombinations in each haplotype block favors the biological observation of haplotype blocks. It is an interesting open problem to design an efficient algorithm for haplotype inferring with the assumption that the evolutionary history of the underlying haplotypes satisfies the galled-tree network model. As we mentioned in the previous section, Gusfield first used the galled-tree model in the context of the haplotyping problem in 2004. However, it was only recently that the full characterization of its existence for haplotype (binary) matrices has been found ([75, 95]). It is still an open problem to find the full-characterization for galled-tree network haplotyping problem.

At the same time, simpler versions of the problem are intriguing to explore as they can provide insights into the original GTNH problem. Song et al ([94]) studied the case where only one recombination is allowed in a galled-tree network and with the assumption that a PPH solution, if there is any, for each sub-matrix (M_L and M_R) of the input genotype matrix is unique. One open problem is to find efficient algorithms for the GTNH problem with the output galled-tree network with exactly one gall and there is no limitation on the number of PPH solutions for each sub-matrix. Furthermore, it is interesting to see what happens to the corresponding GTNH problem when one adds restrictions on the size of each recombination cycle instead of the number of recombinations. Another version of the problem is by adding constraints to the input genotype matrices. If any of these simpler versions of the GTNH problem is found to be NP-hard, then so is the original GTNH problem.

In this thesis, we focus mainly on three aspects of the problem:

- The characterization of the galled-tree network's existence for binary matrices;

- The combinatorial algorithm for some special case of the galled-tree network haplotyping problem;
- The characterization of the galled-tree network haplotyping problem.

The galled-tree network haplotyping problem is an interesting and yet difficult problem. The characterization of the problem remained open for several years after it was introduced in 2004 ([54]). Our work is among the first to characterize the galled-tree network's existence for binary matrices. We are also the first to show the complexity of the general GTNH problem. Finally, we found polynomial algorithm for the GTNH problem with small galls. Evaluation on simulation data sets shows encouraging results.

The organization of the thesis is as follows: in Chapter 2, we give preliminary definitions for galled-tree network, galled-tree network haplotyping problem as well as review related work; in Chapter 3, we describe the characterization of the galled-tree network's existence for binary matrices; in Chapter 4, we describe a special case of the GTNH problem and the corresponding polynomial algorithm that solves the problem; in Chapter 5, we describe another special case of the GTNH problem and its characterization. Furthermore, we show that the GTNH problem is NP-complete. Note that in attempting to find the full characterization of the GTNH problem, we found some results in ordered graph coloring problems. However, these results are independent with the results for the haplotyping problem and are omitted from this thesis. Interested readers are referred to ([45]) for references.

Chapter 2

Galled-Tree Networks

In this chapter, we give preliminary definitions for galled-tree network, galled-tree network haplotyping problem and review the results regarding the characterization, construction and application of galled-tree networks.

2.1 Definitions

Definition 7. A *phylogenetic network* N on m characters is a directed acyclic graph containing exactly one vertex (the root) with no incoming edges. Each vertex other than the root has either one or two incoming edges. Where there is 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.*

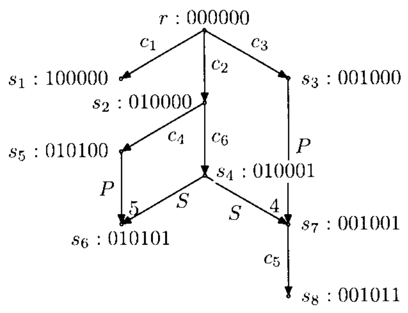
Each integer (character) from 1 to m is assigned to exactly one mutation edge in N and each mutation edge is assigned one character. Each vertex in N is labeled by a binary sequence of length m, starting with the root vertex which is labeled with the all-0 sequence. Since N is acyclic, the vertices in 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:

(1) 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.

(2) 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.

Example 1. A phylogenetic network for a given binary matrix M is illustrated in Figure 2.1.



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 2.1: 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 8. Given an $n \times m$ binary matrix A, we say that a phylogenetic network N with m characters *explains* A if each sequence of A is a label of some vertex in N.

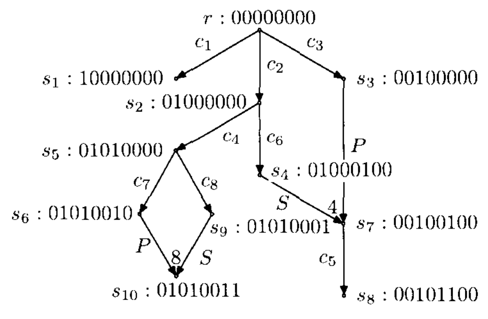
Definition 9. (Galled-tree network) In a phylogenetic network N, let v be a vertex that has two paths out of it that meet at a recombination vertex x (v is the lowest common ancestor of the parents of x). The two paths together form a *recombination cycle* Q. The vertex v is called the *coalescent vertex*. We say that Q contains a character c, if c labels one of the mutation edges of Q.

A phylogenetic network is called a *galled-tree network* if no two recombination cycles share vertices. A recombination cycle of a galled-tree network is sometimes referred to as a *gall*. A galled-tree network is called *reduced galled-tree* if every gall contains at least one conflicted pair of sites, and contains no unconflicted sites.

Example 2. A galled-tree network for a binary matrix M is illustrated in Figure 2.2. Note that the two recombination cycles in the network do not share any edges.

Given an $n \times m$ matrix A with values in $\{0, 1\}$, we say that a galled-tree network N with m characters *explains* A if each sequence of A is a label of some vertex in N.

Definition 10. Given a genotype matrix A, we say that A can be *explained* by a galled-tree network if there exists a haplotype matrix B inferred from A such that B can be explained by a galled-tree network.



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 2.2: 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.

By the definition of galled-tree network, the following observation follows:

Observation 1. *Given a haplotype matrix M, let M' be a matrix obtained from M by duplicating one row or adding a row with all-0 sequence. Matrix M' can be explained by a galled-tree network N if and only if M can be explained by N. Let M'' be a matrix obtained from M by removing one row. If M can be explained by a galled-tree network N then M'' can also be explained by N.*

Problem 3. (Galled-Tree Network Haplotyping problem) Given a genotype matrix A, decide if A can be explained by a galled-tree network.

The algorithm for constructing galled-tree network for a binary matrix uses the concept of conflict character and conflict graph.

Definition 11. (Conflicting Characters) Given an $n \times m$ binary matrix A, two characters/columns c and c' *conflict* in A if $A[c, c']$ contains three rows with pairs $[0, 1], [1, 0]$ and $[1, 1]$. A character is *unconflicted* if it does not conflict with any other character of A.

Note that in this thesis, we assume the root to be the all-0 vector, unless otherwise specified.

Definition 12. Given an $n \times m$ binary matrix A, the *conflict graph* G_A of A has the vertex set $\{1, \ldots, m\}$ and for every two characters c and c', (c, c') is an (undirected) edge of G_A if and only if they conflict.

2.2 Results for Galled-Tree Networks

2.2.1 Construction of Galled-Tree Networks for Binary Matrices

Gusfield et al. [54] presented a polynomial algorithm for constructing a galled-tree network for binary data (haplotype data in particular). The algorithm is based on a number of necessary conditions on the existence of the network, some of which are properties of the *conflict graph* G. Specifically, they show that there is a one-to-one correspondence between any non-trivial connected components of G and the galls in a reduced galled-tree network N, where a reduced galled-tree network is a galled-tree network with each recombination cycle only containing conflicting sites. Each non-trivial connected component G_C of G is a bipartite graph, with unique bipartition, and indices of vertices (sites) on one partition are strictly smaller than vertices on the other partition. The recombination point r for the gall can be any value in the gap between the indices of the two partitions. Let C be the set of columns correspond to the vertices of G_C. Another important feature is that there is a sequence x in $M[C]$ such that after removing all copies of x from $M[C]$ (let $M[C] - x$ denote the operation) there is a perfect phylogeny T_C for the resulting matrix. T_C is organized into one or two paths with the recombination of the two "end" sequences (from either the root sequence of the prefect phylogeny and the single leaf sequence, or the two leaf sequences) at the recombination point r creates x. The algorithm can be briefly described as follows.

1. The algorithm handles each gall separately. For each non-trivial component G_C of G, the algorithm first determines the recombination point r. Then, it checks the corresponding sub-matrix $M[C]$ and finds its x. If no x exists such that $M[C] - x$ has perfect phylogeny, then there is no galled-tree network for M. Otherwise, for each such x, check if there is a perfect phylogeny for $M[C] - x$ that contains exactly one or two paths and whose end sequences can be recombined at point r to create x. If not, then there is no galled-tree network for M.

2. Find the ancestral relation between any two galls, and get a forest $\tilde{T}$ of galls. Gall Q is an *ancestor* of a gall Q' in a galled tree T if there is a directed path in T from some node on Q to the coalescent node of Q'. This can be determined by checking the matrix M to see if there exists a row Z that contains 1 in both some site from Q and some site from Q'.

3. Extend $\tilde{T}$ by adding in unconflicted sites and placing the sequences of M at specific leaves. In particular, the algorithm first constructs the perfect phylogeny T based on the species that do not contain 1 for any of the conflict characters. Then the

algorithm determines where each gall Q in $\tilde{T}$ resides relative to T by checking rows in M. Rows that have 1 for sites in T and sites on certain gall Q in $\tilde{T}$ help determine Q's position in T. Now, the forest $\tilde{T}$ becomes a galled-tree. The algorithm then proceeds to label leaf vertices with species by doing a bottom-up traversal of T. Species that contain 1 for sites on certain gall should be descendant of some sites on that gall. Finally, the algorithm adds the remaining unconflicted sites onto T.

Note, the procedures to determine the ancestry relation between galls, the ancestry relation between galls and sites on the perfect phylogeny T, and to determine where to insert the remaining unconflicted sites into T are all based on the fact that for any site i, it either mutates on a directed edge (u, v) or is contained by a gall Q and species that are reachable from v through a directed path, which does not contain any edge on Q, has the value 0 or 1 as v has for site i.

Gusfield, Eddhu and Langley later[53] gave a slightly improved algorithm for the galled-tree construction.

2.2.2 Characterization of Galled-Tree Network's Existence

We [75] gave a full characterization of the existence for galled-tree networks for binary matrices. In particular, we showed that some of [54]'s necessary conditions are sufficient for galled-tree networks' existence. The detail of the characterization is described in Chapter 3.

Song [95] showed another full characterization for galled-tree's existence. The characterization is based on the combinatorial constraints induced by every pair of columns in M, and is essentially the same as ours.

2.2.3 More General Phylogenetic Networks

There is work in constructing phylogenetic networks for binary matrices which relax some constraints of the galled-tree networks. In [50], Gusfield studied the case when the root of galled-tree network is unknown. They proved that if there is an optimal (with minimum number of recombinations) galled-tree for input matrix M, then there is an optimal galled-tree network with one of the species in M as its root. This makes the construction of root-unknown galled-trees solvable in polynomial time. The algorithm can also be extended to model multiple-crossover recombinations. In [51], Gusfield et al. studied a more general phylogenetic network, where recombination cycles can share edges. They define ***blob*** to be a maximal set of recombination cycles, where each recombination cycle inside the blob shares at least an edge with some other recombination cycles in the set. Gusfield et al.

conjecture [51] that for any input matrix M, there is always a phylogenetic network N for M with the minimum number of recombinations over all possible phylogenetic networks for M, in which every ***blob*** in N contains all and only the characters from a single connected component of the conflict graph G_M.

2.2.4 Application of Galled-Tree Networks

Galled-tree networks are used in haplotyping problems. The galled-tree network haplotyping problem is analogous to the perfect phylogeny haplotyping problem, with the underlying evolutionary model for the original haplotypes being a galled-tree network. The first problem is much harder. After it was introduced in 2004 ([54]), not much progress has been made on the problem and it is still not fully characterized. Song et al. [94] studied the case when the underlying haplotypes, that form the genotypes, evolved on a galled-tree network with exactly one recombination cycle on it. In [94], Song et al. gave a framework of the algorithm. For an input genotype matrix M, the algorithm splits M column-wise into two sub-matrix M_L and M_R. For each such partition, check if both M_L and M_R have a PPH solution, and if so, check if the two corresponding perfect phylogeny trees T_L and T_R are exactly one SPR (subtree-prune-and-regraft) operation away. If so, there exists a galled-tree network with exactly one gall on it that explains M. Note, Song et al. assume uniqueness of PPH solution for each sub-matrix in each partition of M.

Galled-tree network is studied in other contexts as well. Jansson et al. [66] gave polynomial algorithm for the problem of determining whether a given set of rooted triplets (tree with three distinctively labeled leaves) can be merged into a galled-tree network. Nakhleh et al. [80] studied the problem of combining two trees into a galled tree network, which was claimed helpful in understanding biological events such as hybridization and horizontal gene transfer. Huynh et al. [62] further extended their algorithms so that it can handle the case when the number of input trees is not limited to 2, and there is no restriction for the degree of the input trees.

Chapter 3

Characterization of the Existence of Galled-Tree Networks

In this Chapter, we describe a complete characterization of the existence of a galled-tree network in the form of simple sufficient and necessary conditions for both root-known and root-unknown cases. Following that, a simple algorithm for constructing galled-tree networks is illustrated, as well as a new necessary condition for the existence of a galled-tree network, similar to bi-convexity.

3.1 Characterization of the Existence of a Galled-Tree Network in the Root-Known Case

In this section we give a complete characterization of the existence of a galled-tree network explaining a given $n \times m$ binary matrix A when the root sequence r is known, where n represents the number of rows and m the number of columns of A. Without loss of generality we can assume that $r = 0^m$, since otherwise we can invert every column of A in which r has 1; it is straight-forward to show that and the new matrix can be explained by a galled-tree network with root labeled 0^m if and only if A can be explained by a galled-tree network with root labeled r. Indeed, the two galled-tree networks are isomorphic (have the same structures) and differ only in their labelings and one labeling can be easily converted to another by the inversions that convert r to the all-0 sequence.

To give a complete characterization in the all-0 root case, we show that the two necessary conditions (Lemma 4 and Theorem 10) in [54] are also sufficient.

Our characterization of galled-tree networks is presented in the following theorem.

Theorem 3. *Given an $n \times m$ binary matrix A, there exists a galled-tree network with* *root $r = 0^m$ explaining A if and only if every nontrivial component (having at least two vertices) K of the conflict graph G_A 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; and*

2. *There exists a sequence $x \neq 0^{|K|}$ such that $A[K] - x$ has no conflicting characters.*

Note that we assume the natural order of the set of characters for A and thus it makes sense to say that one character c is smaller than another character c'.

In the rest of this section we prove several results which together with results in [54] will imply the theorem. Let A be a given $n \times m$ binary matrix and let $r = 0^m$.

The following crucial result shows that if condition (2) of Theorem 3 is satisfied then $A[K] - x$ can be explained by a tree with at most two edge-disjoint branches.

Lemma 4. *If a component K of G_A is bipartite with partitions L and R, and $A[K] - x$ has no conflicting characters for some $x \neq 0^{|K|}$, then any phylogenetic tree T explaining $A[K] - x$ has at most two branches. For $i = 0, 1$, let L_i (R_i) be the set of all $c \in L$ ($c \in R$) such that $x[c] = i$. One possible branch contains all edges labeled with characters in $L_1 \cup R_0$, and the other contains all edges labeled with characters in $R_1 \cup L_0$. If T has two branches then they do not share any edge (recall that we assume that a phylogenetic tree has all edges labeled by characters).*

Before proving the lemma, we state the following observation on ancestry relations between characters. We gave a well-known theorem of perfect phylogenies first, which is the basis for the observation.

Theorem 5. *Given an $n \times m$ binary matrix A, there exists a perfect phylogenetic tree explaining A if and only if no two characters conflict in A.*

Note that if we drop the requirement in the definition of perfect phylogeny trees to have the root labeled with the all-0 sequence, the above theorem is still true, although we have to redefine conflicts between characters: c and c' conflict in A if $A[c, c']$ contains all the 4 possible value pairs $[0, 0]$, $[0, 1]$, $[1, 0]$ and $[1, 1]$ (the *four-gamete test*).

Observation 2. *Given a binary matrix M and a perfect phylogenetic tree T explaining M, for any two characters c and c',*

- *if $M[c, c']$ contains pairs $[0, 1]$ and $[1, 1]$ then $e(c') \preceq e(c)$ (contains pairs $[1, 0]$ and $[1, 1]$ then $e(c) \preceq e(c')$), hence $e(c)$ and $e(c')$ are comparable;*

- *if $M[c, c']$ contains only the pairs $[0, 0]$ and $[1, 1]$ then $e(c)$ and $e(c')$ are comparable; moreover, for any other edge e (with label d) on the shortest path containing edges $e(c)$ and $e(c')$ in T, $M[c, c', d]$ contains only the triples $[0, 0, 0]$ and $[1, 1, 1]$;*

- *if $M[c, c']$ contains pairs $[0, 1]$ and $[1, 0]$ then $e(c)$ and $e(c')$ are incomparable.*

Note that no other case is possible as M does not contain any conflict and the general assumption on M is that it does not contain any all-0 columns.

Proof of Lemma 4. Let L and R be the two bipartitions of K. For $i = 0, 1$, let L_i (R_i) be the set of all $c \in L$ ($c \in R$) such that $x[c] = i$. Let T be a perfect phylogenetic tree for $A[K] - x$. Note that the removal of x from $A[K]$ eliminates all conflicts in K. Since, every $c \in K$ is connected to some other $c' \in K$, there is another row (not containing sequence x) having state 1 for character c. Hence, each column in $A[K] - x$ contains 1. We will use this fact several times.

Claim 1. *Edges $\{e(c)\}_{c \in L_1}$ are pairwise comparable, i.e., they lie on one directed path in T. The same holds for edges in $\{e(c)\}_{c \in R_1}$.*

Proof. Let $c, c' \in L_1$. Since c and c' lies in the same partition there is no conflict between them in $A[K]$. Since $A[c, c']$ contains pair $[1, 1]$ (in x), $A[c, c']$ cannot contain both pairs $[1, 0]$ and $[0, 1]$. By Observation 2, edges $e(c)$ and $e(c')$ are comparable. □

Claim 2. *G_A does not contain any edge (c, c') where $c \in L_0$ and $c' \in R_0$.*

Proof. Let (c, c') be a conflict in $A[K]$ such that $c \in L_0$ and $c' \in R_0$. Since $x[c, c']$ is $[0, 0]$, c and c' conflict also in $A[K] - x$, a contradiction. □

Consequently, since K is connected, we can assume that for every $c \in L_0$ (respectively, $c \in R_0$) there is a $c' \in R_1$ (respectively, $c' \in L_1$) such that c and c' conflict in A. For each such conflict we have:

Claim 3. *For every conflicting characters $c \in L_0$ ($c \in R_0$) and $c' \in R_1$ ($c' \in L_1$) in A, $e(c) \preceq e(c')$.*

Proof. Since c, c' conflict in $A[K]$, $A[c, c']$ contains all pairs $[0, 1]$, $[1, 0]$ and $[1, 1]$. After the removal of all rows containing x from $A[K]$, since $x[c, c']$ is $[0, 1]$, the pairs $[1, 0]$ and $[1, 1]$ are still contained in $(A[K] - x)[c, c']$. By Observation 2, we have that $e(c) \preceq e(c')$. □

By Claim 1, edges in $\{e(c)\}_{c \in R_1}$ lie on one directed path starting at the root of T. By Claim 2, every $e(c) \in L_0$ is connected to some $c' \in R_1$, and hence by Claim 3, it must lie on the same directed path. Consequently, we have that edges $\{e(c)\}_{c \in L_0 \cup R_1}$ (respectively, edges $\{e(c)\}_{c \in R_0 \cup L_1}$) lie on one directed path in T. Obviously, if one of the sets $L_0 \cup R_1$ and $R_0 \cup L_1$ is empty, then T has only one branch and we are done. In what follows we will show that if both these paths are non-empty, then they do not share any edges.

Claim 4. *For every two characters $c \in L_1$ and $c' \in R_1$, if there is an edge (c, c') in G_A then $e(c)$ and $e(c')$ are incomparable.*

Proof. Since c, c' conflict in $A[K]$, $A[c, c']$ contains all pairs $[0,1]$, $[1,0]$ and $[1,1]$. After the removal of all rows containing x from $A[K]$, since $x[c, c']$ is $[1,1]$, the pairs $[0,1]$ and $[1,0]$ are still contained in $(A[K] - x)[c, c']$. By Observation 2, we have that $e(c)$ and $e(c')$ are incomparable. □

Claim 5. *For every two characters $c \in L_0$ ($c \in R_0$) and $c' \in L_1$ ($c \in R_1$), either $e(c)$ and $e(c')$ are incomparable or $e(c') \preceq e(c)$ or $(A[K]-x)[c, c']$ contains only pairs $[0,0]$ and $[1,1]$ (i.e., the columns c and c' are identical in $A[K] - x$).*

Proof. Since c and c' lie in the same partition, they do not conflict in $A[K]$. Since $x[c, c']$ is $[0,1]$, $A[c, c']$ cannot contain both pairs $[1,0]$ and $[1,1]$. Since $A[K] - x$ contains 1 in column c, $(A[K] - x)[c, c']$ must contain at least one of the pairs $[1,0]$ and $[1,1]$. If it contains $[1,0]$ then since also column c' contains 1, $(A[K] - x)[c, c']$ must contain also pair $[0,1]$. Then, by Observation 2, c and c' are incomparable. If it contains $[1,1]$ and also $[0,1]$ then $e(c') \preceq e(c)$. Otherwise, it contains only pairs $[0,0]$ and $[1,1]$. □

Claim 6. *For every two characters $c \in L_1$ and $c' \in R_1$, there is an edge (c, c') in G_A.*

Proof. Assume for contradiction that c and c' are not connected by an edge in G_A. Since $x[c, c']$ is $[1,1]$, $A[c, c']$ cannot contain both pairs $[0,1]$ and $[1,0]$. Consequently, since $A[K] - x$ contains 1's in both columns c and c', $(A[K] - x)[c, c']$ must contain the pair $[1,1]$. First, assume it contains only pairs $[0,0]$ and $[1,1]$. Then the columns c and c' are identical in A, hence they must have the same set of neighbors in the conflict graph G_A. This is a contradiction, since they lie in different partitions of the bipartite graph K.

Therefore, we can assume that $(A[K] - x)[c, c']$ contains also exactly one of the pairs $[0,1]$ and $[1,0]$. Without loss of generality, assume that it is $[0,1]$. By Observation 2, $e(c') \preceq e(c)$. Since, K is connected, c' must have a neighbor in $L_0 \cup L_1$. First, assume it has a neighbor $c'' \in L_1$. By Claim 1, c and c'' lie on the same directed path starting in the root. Since $e(c') \preceq e(c)$, c' lies on this path too. By Claim 4, c' and c'' are incomparable, a contradiction. Second, assume c' has a neighbor $c'' \in L_0$. By Claim 3, $e(c'') \preceq e(c')$. We have $e(c'') \preceq e(c') \preceq e(c)$. By Claim 5, this is possible only if $(A[K] - x)[c, c'']$ contains only pairs $[0,0]$ and $[1,1]$. By Observation 2, $(A[K]-x)[c,c']$ contains only pairs $[0,0]$ and $[1,1]$ as well, a contradiction with the fact that $e(c') \preceq e(c)$. □

It follows by Claims 4 and 6 that any edge in $\{e(c)\}_{c \in L_1}$ is incomparable with any edge in $\{e(c)\}_{c \in R_1}$. To finalize the separation of the path containing edges in $\{e(c)\}_{c \in L_0 \cup R_1}$ from the path containing edges in $\{e(c)\}_{c \in R_0 \cup L_1}$ we need to strengthen Claim 5.

Claim 7. *For every two characters $c \in L_0$ ($c \in R_0$) and $c' \in L_1$ ($c \in R_1$), $e(c)$ and $e(c')$ are incomparable.*

Proof. Consider two characters $c \in L_0$ and $c' \in L_1$. Since K is connected, there is a neighbor $c'' \in R_1$ of c in K. By Claim 3, $e(c) \preceq e(c'')$. By Claims 6 and 4, $e(c')$ and $e(c'')$ are incomparable. Then either $e(c) \preceq e(c')$, or $e(c)$ and $e(c')$ are incomparable. The first case is not possible by Claim 5. □

By the above claims it follows that the edges in $\{e(c)\}_{c \in L_0 \cup R_1}$ lie on one directed path, and no edge in $\{e(c)\}_{c \in R_0 \cup L_1}$ lies on this path, and vice versa. Hence, if these two paths share any common edge, such an edge cannot be labeled by any character in K, i.e., it must be unlabeled which is the contradiction with the definition of the galled-tree networks. □

In the proof of Lemma 4, we proved several important properties related to the structure of the component K of the conflict graph to the sequence x. We summarize them as follows:

Lemma 6. *Assume that a component K of G_A is bipartite and $A[K]-x$ has no conflicting characters for some $x \neq 0^{|K|}$. Let L and R be the two bipartitions of K. For $i = 0, 1$, let L_i (R_i) be the set of all $c \in L$ ($c \in R$) such that $x[c] = i$. Then K does not contain any edge between sets L_0 and R_0, and a subgraph of K induced by vertices $L_1 \cup R_1$ is a complete bipartite graph.*

In the following theorem we show that if a component of the conflict graph G_A satisfies both conditions of Theorem 3 then there is a gall explaining $A[K]$.

Theorem 7. *If a component K of G_A is bipartite with partitions L and R, $A[K]-x$ has no conflicting characters for some $x \neq 0^{|K|}$ and all vertices in L are smaller than all vertices in R, then $A[K]$ can be explained by a galled tree containing one recombination cycle (gall) rooted in the node with label $0^{|K|}$ and having x as the label of the recombination vertex.*

Proof. By Lemma 4, there is a phylogenetic tree T explaining $A[K] - x$ with at most two branches. Let B_P be the branch containing edges labeled with characters in $L_1 \cup R_0$, and B_S the branch containing edges labeled with characters in $R_1 \cup L_0$. If one of these two sets is empty then one of the branches is empty as well. Furthermore, the vertex labeled $0^{|K|}$ is the only vertex shared by B_P and B_S. Now, we will add a recombination vertex z into T. Let y_P (y_S) be the last vertex on the branch B_P (B_S). Add two recombination edges (y_P, z) labeled P and (y_S, z) labeled S, cf. Figure 3.1. Set the recombination point r_z to any character in $\{p+1, \ldots, q\}$, where p is the maximum character in L and q is the minimum character in R. We will show that the label of recombination vertex z is x, i.e., the gall explains the matrix $A[K]$.

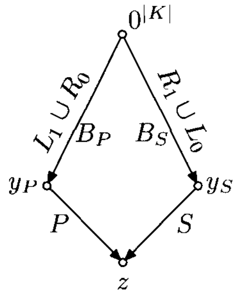


Figure 3.1: Construction of recombination cycle using two branches B_P and B_S of the perfect phylogenetic tree for $A[K] - x$.

The label of z is formed by concatenating the first $r_z - 1$ characters of $P(z)$ (see Definition 7) with the last $|K| - r_z + 1$ characters of S_z. The label $P(z)$ (respectively, $S(z)$) has 0 (respectively, 1) in every position $c \in R_1 \cup L_0$ and 1 (respectively, 0) in every position $c \in L_1 \cup R_0$. The label of z at position $c \in L_0$ comes from $P(z)$, hence it has value 0. Similar arguments show that the label of z agrees with x also on all remaining positions, as required. □

In the following we define a compressed matrix which will be used to build a phylogenetic network. Note that the compressed matrix is similar to the pass-through matrix [52]. However, the pass-through matrix does not contain columns for components of the conflict graph which are singletons.

Definition 13. Let $K_1, \ldots, K_k$ be the components of the conflict graph G_A. The *compressed matrix* C_A is the $n \times k$ binary matrix with columns labeled by $K_1, \ldots, K_k$. It has 1 in row $i \in \{1, \ldots, n\}$ and column K_j, $j \in \{1, \ldots, k\}$, if and only if the row i in $A[K_j]$ contains at least one 1.

Lemma 8. *The compressed matrix C_A has no conflicting characters.*

Proof. Let K_j and $K_{j'}$ be two components of G_A. Suppose by a way of contradiction K_j and $K_{j'}$ conflict in C_A, i.e., there are rows r_1, r_2, r_3 containing pairs $[1,1], [1,0], [0,1]$ in $C_A[K_j, K_{j'}]$, respectively. By definition of C_A, there are characters $c \in K_j$ and $d \in K_{j'}$ such that the matrix A has the pair $[1,1]$ in the row r_1. Suppose first that columns c and d are identical in A. Then they have to have the same neighborhood in the conflict graph. Since, they are in different components, these two components must be trivial (singletons). However, this implies that the columns K_j and $K_{j'}$ are identical in C_A, a contradiction. Hence, by Observation 2, $A[c,d]$ is either 01-free or 10-free.

Assume that $A[c,d]$ is 10-free. The other case is symmetric. In the following claim we will show that for every $\bar{c} \in K_j$, $A[\bar{c},d]$ is also 10-free.

Claim 8. *For every $c \in K_j$ and $d \in K_{j'}$,*

- *if $A[c,d]$ is 10-free then $A[\bar{c},d]$ is also 10-free for every $\bar{c} \in K_j$;*
- *if $A[c,d]$ is 01-free then $A[c,\bar{d}]$ is also 01-free for every $\bar{d} \in K_{j'}$.*

Proof. We will show only the first part of the claim. The second follows by symmetry. If c is the only vertex in K_j, we are done. Otherwise, let c' be a neighbor of c, i.e., c and c' conflict. Hence, there are rows r_1 and r_2 containing pairs $[1,1]$ and $[1,0]$, respectively, in $A[c,c']$. Since $A[c,d]$ is 10-free, the two rows must contain 1 in $A[d]$. Since $A[c',d]$ contains pairs $[1,1]$ and $[0,1]$ in rows r_1 and r_2, and c' and d do not conflict, $A[c',d]$ is 10-free. By applying the argument recursively on remaining characters in K_j, the claim follows. $\square$

Now, let us consider row r_2. Since r_2 contains the pair $[1,0]$ in C_A, by definition of C_A, there is a character $c' \in K_j$ such that $A[c',d]$ has the pair $[1,0]$ in this row. This contradicts the fact that $A[c',d]$ is 10-free and proves the lemma. $\square$

It follows that the compressed matrix C_A can be explained by a perfect phylogenetic tree. We will use this tree to construct the galled-tree network explaining A. Recall that a perfect phylogenetic tree with a fixed root is unique up to order of edges labeled with characters having identical columns in the input matrix. From all phylogenetic trees explaining C_A we want to pick one satisfying the following condition:

Definition 14. A phylogenetic tree T explaining C_A is called *sorted* if for every two identical columns K_j and $K_{j'}$ such that component K_j is a singleton and component $K_{j'}$ has at least two vertices in the conflict graph, $e(K_j) \prec e(K_{j'})$.

The following lemma shows that sequences in rows of A behave nicely with respect to edges in a sorted phylogenetic tree T explaining the compressed matrix C_A.

Lemma 9. *Let T be a sorted phylogenetic tree explaining the compressed matrix C_A. Assume that $e(K_j) \prec e(K_{j'})$ in T for some components K_j and $K_{j'}$ in G_A. Consider all rows containing a 1 in $A[K_{j'}]$, i.e., having 1 in $C_A[K_{j'}]$. Then all sequences in these rows in $A[K_j]$ are identical and different from the all-0 sequence.*

Proof. Since $e(K_j) \preceq e(K_{j'})$, $C_A[K_j, K_{j'}]$ is 01-free. Obviously, for every row having 1 in $C_A[K_{j'}]$, the sequence in $A[K_j]$ cannot be the all-0 sequence. We will show that those sequences over all such rows are identical. Suppose by a way of contradiction there exist rows r_1 and r_2 and characters $c \in K_j$ and $d, d' \in K_{j'}$ such that $A[c,d]$ contains pair $[1,1]$ in row r_1 and $A[c,d']$ contains pair $[0,1]$ in row r_2. Since, c and d are in different components, $A[c,d]$ is either 01-free or 10-free.

First, assume $A[c,d]$ is 01-free. By Claim 8, it follows that $A[c,d']$ is also 01-free, a contradiction.

Second, assume that $A[c,d]$ is 10-free. By Claim 8, for every character $c' \in K_j$, $A[c',d]$ is also 10-free. If $K_{j'}$ is a singleton, then since T is sorted, either columns K_j and $K_{j'}$ are not identical in C_A, or K_j is singleton as well. In the first case, since $C_A[K_j, K_{j'}]$ is 01-free, they contain the pair $[1,0]$. Therefore, there is a row r_3 which has the pair $[1,0]$ for some column $c' \in K_j$ and d in the matrix A. This is a contradiction since $A[c',d]$ is 10-free. In the second case, the columns K_j and $K_{j'}$ in C_A and columns c and d in A are all identical. Hence, all rows containing 1 in $C_A[K_{j'}]$, have $A[K_j] = [1]$, and the claim follows.

Now, assume that $K_{j'}$ is not a singleton. Let $d'' \in K_{j'}$ be a character conflicting d. It follows that there is the pair $[1,0]$ in $A[d'',d]$ in row r. Since for every character $c' \in K_j$, $A[c',d]$ is also 10-free, $A[K_j]$ contains only 0's in r. Hence, $C_A[K_j, K_{j'}]$ contains $[0,1]$ in r, a contradiction. □

The following algorithm constructs a galled-tree network N_A from a sorted phylogenetic tree for C_A.

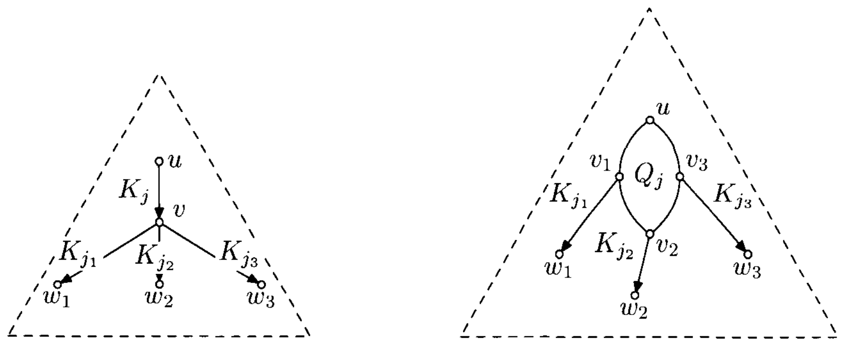


Figure 3.2: Replacing an edge labeled K_j with a gall Q_j.

Algorithm 1.

Input: An $n \times m$ binary matrix A satisfying assumptions of Theorem 7.

1. Construct a sorted phylogenetic tree T of C_A and for every component K_j, $j \in \{1, \ldots, k\}$, of G_A, construct the gall Q_j explaining $A[K_j]$.

2. In top-down fashion process every edge (u, v) labeled K_j. If K_j is a singleton, i.e., $K_j = \{c\}$, replace the label of (u, v) by c. Otherwise, replace the edge with a gall Q_j for K_j as follows (cf. Figure 3.2):

   2.1 Remove edge (u, v).

   2.2 Identify the coalescent node of the gall Q_j with u.

   2.3 For every edge (v, w) labeled $K_{j'}$, consider any row r containing 1 in $C_A[K_{j'}]$. Let s be the sequence in $A[K_j]$ in row r. By Lemma 9, $s \neq 0^{|K_j|}$. Since Q_j explains $A[K_j]$, it contains a vertex $v' \neq u$ labeled s. Remove the edge (v, w), add the edge (v', w) and label it $K_{j'}$.

   2.4 Remove vertex v.

3. To obtain a proper labeling of vertices in N_A, compute new labels of length m using the procedure described in the definition of galled-trees.

The following lemma shows that the algorithm produces essentially unique answer. More precisely,

Lemma 10. *After constructing a sorted phylogenetic tree T of C_A and galls Q_j's for every component K_j of G_A in Step 1 of Algorithm 1, the remaining construction of the algorithm produces unique result (the resulting galled-tree network depends only on selection of T and Q_j's).*

Proof. The only choice we have in the remaining steps of the algorithm is in Step 2.3 when we can choose any row r containing 1 in $C_A[K_{j'}]$. The selection of vertex v' to which we attach w depends on the sequence s in row r of the matrix $A[K_j]$. However, by Lemma 9, for every row r' containing 1 in $C_A[K_{j'}]$, the sequence in row r' of the matrix $A[K_j]$ is also s. □

The question of how many different galls are there for a matrix $A[K_j]$ was studied by [54]. It was shown that there are at most three different galls, and if there are enough characters in K_j, there is only one gall explaining $A[K_j]$. Also note that the phylogenetic tree T is unique up to arrangement of characters with identical columns on edges. For our purposes, the fact that Step 2.3 can be performed only in one unique way is sufficient to show that N_A explains A.

Theorem 11. *Assume that every non-trivial (with at least two vertices) component K of G_A is bipartite with partitions L and R, $A[K] - x$ has no conflicting characters for some $x \neq 0^{|K|}$ and all vertices in L are smaller than all vertices in R. Then the galled-tree network N_A constructed in Algorithm 1 explains A.*

Proof. Let T be a sorted phylogenetic tree explaining C_A used to build the galled-tree network N_A. Let $K_1, \ldots, K_k$ be the components of the conflict graph G_A in the order they were processed in Algorithm 1. It is sufficient to show that for every row r in A, the sequence s in this row appears in N_A.

We will define a sequence of $k+1$ matrices $C_A = M_0, M_1, \ldots, M_k = A$ and a sequence of $k+1$ galled-tree networks $T = N_0, N_1, \ldots, N_k = N_A$. Note that N_j explains M_j when $j = 0$. We will prove that this is so for every $j \in \{0, \ldots, k\}$ by induction on j. The theorem will follow. Let us start with definition of the two sequences.

The matrix M_j is a concatenation of $C_A[K_{j+1}, \ldots, K_k]$ and $A[K_1 \cup \cdots \cup K_j]$, i.e.,

$$M_j = \Big(C_A[K_{j+1}, \ldots, K_k] \quad A[K_1 \cup \cdots \cup K_j]\Big) \, .$$

The matrix M_j can be also obtained from M_{j-1} by removing its first column K_j and inserting columns of $A[K_j]$ inside of $A[K_1 \cup \cdots \cup K_{j-1}]$ to their correct positions (preserving relative order in A).

Note that since A does not contain any all-0 column, neither does C_A, and hence T contains an edge for every label K_j. The network N_j is constructed from N_{j-1} by processing an edge labeled K_j as described in Algorithm 1 and updating all labels of vertices. The new network is defined on the set of characters of M_j, i.e., instead of character K_j it has characters contained in component K_j, and the order of the new set of characters (on each label) is consistent with the order of columns in M_j.

Assume that N_{j-1} explains M_{j-1}. We will show that also N_j explains M_j. First, assume that $K_j = \{c\}$ is a singleton. Take a sequence s in row r in M_j and the sequence s' in the same row in M_{j-1}. By induction, there is a vertex v in N_{j-1} labeled s'. Consider the label of the vertex v in N_j. Since the order of character labels in N_i and columns in M_i, $i = \{j-1, j\}$, is the same; the column K_j in M_{j-1} is identical to column c in M_j; and the edge labeled K_j in N_{j-1} is relabeled to c in N_j, the label of v is s.

Second, assume that K_j has at least two elements. Let (u, v) be the edge in N_{j-1} labeled K_j. Take a sequence s in row r in M_j and the sequence s' in the same row in M_{j-1}. By induction, there is a vertex z in N_{j-1} labeled s'. If $s'[K_j] = 0$, then z is not a vertex in the subtree of N_{j-1} rooted at v (recall by order of components, this is indeed a tree). For every $c \in K_j$, $s[c] = 0$ and $s[c]$ for the remaining characters is the same as $s'[c]$. Let l be the label of z in N_j. It is easy to see that for every character $c \notin K_j$, $l[c]$ is the same as $s'[c]$. Since, z in N_j is not a vertex in $Q_j - u$, neither a vertex in any of the subtrees attached to Q_j, for every $c \in K_j$, $l[c] = 0$. Hence, $l = s$.

Now, assume that $s'[K_j] = 1$, i.e., z lies in the subtree of N_{j-1} rooted at v. First, assume that $z = v$. The sequence $s[K_j]$ is a row in the matrix $A[K_j]$. Since Q_j explains $A[K_j]$, there exists a vertex $v' \in Q_j$ with label $s[K_j]$. Let l be the label of v' in N_j. Obviously, for every $c \in K_j$, $l[c] = s[c]$. On the other hand, for the remaining characters $c \notin K_j$, $l[c] = s'[c] = s[c]$. Hence, again $l = s$.

Finally, assume that z lies in a subtree rooted at w, where w is a child vertex of v in N_{j-1}. Let $K_{j'}$ be the label of the edge (v, w) in N_{j-1}. In N_j, w is connected to a vertex $v' \in Q_j$. By Lemma 10, we can assume that v' was chosen in Step 2.3 of the algorithm using the row r. Indeed, since z with label s' lies in the subtree rooted in w in N_{j-1}, $s'[K_{j'}] = 1$ and as before, obviously, s and s' agree on all characters $c \notin K_j$. By the construction, the label l' of v' in Q_j is $s[K_j]$, and hence also for every $c \in K_j$, $l'[c] = s[c]$. Since z lies in the subtree rooted at v' in N_j, the same hold also for label l of z in N_j, i.e., for every $c \in K_j$, $l[c] = l'[c] = s[c]$. For the remaining character $c \notin K_j$, as before we have $l[c] = s'[c] = s[c]$. Hence, $l = s$. □

It is known that the number of galls in any galled-tree network explaining A is at least the number of non-trivial components in the conflict graph G_A [54]. Since the galled-tree network constructed by Algorithm 1 has exactly this number of galls, the constructed network is optimal.

Obviously, by Theorem 7, Algorithm 1 cannot fail to construct a galled-tree network N_A, and by the above theorem, the constructed network explains A. Hence, we have the following corollary.

Corollary 1. *If every non-trivial component K of G_A is bipartite with partitions L and R, $A[K] - x$ has no conflicting characters for some $x \neq 0^{|K|}$ and all vertices in L are smaller than all vertices in R, then there exists a galled-tree network explaining A.*

Combining the above corollary with the results of [54], Theorem 3 follows.

In the case $r \neq 0^m$, based on the discussion at the beginning of this section, there is a galled-tree network rooted at r if and only if the inverted matrix satisfies condition of Theorem 3. To formulate the characterization directly for A, we need to slightly redefine the concept of conflict graph, referred to as *incompatibility graph.*

Definition 15. (Incompatible characters) Given an $n \times m$ binary matrix A, two characters/columns c and c' are *incompatible* in A if $A[c, c']$ contains four rows with all four possible pairs.

Definition 16. (Incompatibility graph) Given an $n \times m$ binary matrix A, the *incompatibility graph* H_A has the vertex set $\{1, \ldots, m\}$ and an edge between any pair of incompatible characters.

In the following, by $A+r$ we mean the matrix obtained from A by appending the row r. Note that the incompatibility graph of $A+0^m$ is the conflict graph of A, and consequently, the incompatibility graph of $A+r$ is the conflict graph of the matrix obtained by inverting A with respect to r. Hence, we have the following corollary.

Corollary 2. *Given an $n \times m$ binary matrix A, there exists a galled-tree network with the root labeled $r \in \{0,1\}^m$ explaining A if and only if every nontrivial component K of the incompatibility graph H_{A+r} 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; and*
2. *there exists a sequence $x_K \neq r[K]$ such that $(A+r)[K] - x_K$ has no incompatible characters.*

3.2 Characterization of the Existence of a Galled-Tree Network in the Root-Unknown Case

In this section we will give a complete characterization of the existence of a galled-tree network explaining a given matrix A when the root sequence r is unknown. The characterization is in fact a simple consequence of Corollary 2 since it is enough to find a sequence r for which the conditions of the corollary are satisfied, and conversely, if such a galled-tree network exists then such r must exists by the corollary.

Theorem 12. *Given an $n \times m$ binary matrix A, there exists a galled-tree network explaining A if and only if there exist $r \in \{0,1\}^m$ such that for every nontrivial component K of the incompatibility graph H_{A+r} 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; and*
2. *there exists a sequence $x_K \neq r[K]$ such that $(A+r)[K] - x_K$ has no incompatible characters.*

In [50], it is shown that if there exists a galled-tree network for a matrix A then there exists one with root labeled by a row of A. This result allows us to reformulate the above theorem in a more concise way as follows.

Corollary 3. *Given an $n \times m$ binary matrix A, there exists a galled-tree network explaining A if and only if there exists a row $r \in A$ such that for every nontrivial component K of the incompatibility graph H_A 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; and*

2. *There exists a sequence $x_K \neq r[K]$ such that $A[K] - x_K$ has no incompatible characters.*

3.3 Bi-Inclusiveness

[54] introduced an interesting necessary condition for the existence of a galled-tree network, called *bi-convexity.*

Definition 17. A bipartite graph K with partitions L and R is *convex for R* if the vertices in R can be ordered so that, for each vertex $i \in L$, $N(i)$ forms a closed interval in R. That is, i is adjacent to j and $j' > j$ in R if and only if i is adjacent to all vertices in the set $\{j, \ldots, j'\}$. A bipartite graph is called *bi-convex* if sets L and R can be ordered so that it is simultaneously convex for L and convex for R.

They used bi-convexity to design a fast algorithm for the *site consistency problem* for a matrix A if there is a galled-tree network explaining A. The site consistency problem for a matrix A is to find a minimum number of columns whose removal from A results in a perfect phylogeny. The problem was introduced and shown to be NP-complete [21]. The problem reduces to finding a minimum vertex cover in the conflict graph G_A. For bipartite graphs, the vertex cover can be found in polynomial time and for bi-convex graphs in $O(m^2)$ time (recall that m is the number of vertices in the conflict graph) [27]. It was conjectured by [54] that to find a minimum vertex cover of a bi-convex graph can be done in linear time. We present a new necessary condition, *bi-inclusiveness*, which is stronger than bi-convexity (it implies bi-convexity but not other way round) and observe that the minimum vertex cover of a bi-inclusive graph can be found in linear time.

Definition 18. We say that a collection of sets forms a chain, if there is an order $S_1, \ldots, S_k$ of sets such that $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_k$. A bipartite graph K with partitions L and R is *bi-inclusive* if the sets $N(x_1), \ldots, N(x_k)$ form a chain, where $N(x)$ denotes the neighborhood of x, $L = \{x_1, \ldots, x_k\}$.

Note that it is easy to check that the swapping of partitions does not change the property whether K is bi-inclusive or not.

The next theorem shows that if a matrix A satisfies the sufficient and necessary conditions of Theorem 3, i.e., A can be explained by a galled-tree network, then every component of the conflict graph G_A is bi-inclusive.

Theorem 13. *Given an $n \times m$ binary matrix A, if a component K of G_A is bipartite and $A[K] - x$ has no conflicting characters for some $x \neq 0^{|K|}$, then K is bi-inclusive.*

Proof. By Lemma 6, for every $c \in L_0$, $N(c) \subseteq R_1$ and for every $c \in L_1$, $R_1 \subseteq N(c)$. Therefore, it is enough to show that sets $\{N(c)\}_{c \in L_0}$ (respectively, sets $\{N(c)\}_{c \in L_1}$) form a chain.

Consider $c, c' \in L_0$ such that $e(c) \preceq e(c')$. We will show that $N(c') \subseteq N(c)$. For this, let $d \in R_1$ be such that c' and d conflict. Since $x[c', d]$ is the pair $[0, 1]$, there are rows r_1 and r_2 in $(A[K] - x)[c', d]$ containing pairs $[1, 0]$ and $[1, 1]$, respectively. Since $e(c) \preceq e(c')$, the rows r_1 and r_2 in $A[c, c', d]$ have triples $[1, 1, 0]$ and $[1, 1, 1]$, respectively. Since, $x[c, d]$ is the pair $[0, 1]$, characters c and d are in conflict as well, cf. Figure 3.3 (a). Since, edges in $\{e(c)\}_{L_0}$ lie on one directed path, the corresponding sets $\{N(c)\}_{c \in L_0}$ form a chain.

$$\begin{array}{c|cccl} & c & c' & d & \\ \hline & \mathbf{0} & \mathbf{0} & \mathbf{1} & x \\ r_1 & 1 & 1 & 0 & \\ r_2 & 1 & 1 & 1 & \end{array} \qquad \qquad \begin{array}{c|cccl} & c & c' & d & \\ \hline & \mathbf{1} & \mathbf{1} & \mathbf{0} & x \\ r_1 & 0 & 0 & 1 & \\ r_2 & 1 & 0/1 & 1 & \\ r_3 & 1 & 1 & 1 & \end{array}$$

(a) (b)

Figure 3.3: The submatrix $A[c, c', d]$ showing that: (a) if $c, c' \in L_0$: if c' conflicts with d then so does c; (b) if $c, c' \in L_1$: if c conflicts with d then so does c'.

Now, consider $c, c' \in L_1$ such that $e(c) \preceq e(c')$. We will show that $N(c) \subseteq N(c')$. Since, both sets contain R_1 as a subset, it is enough to show that for every $d \in R_0$ if c and d conflict then also c' and d conflict. Since $x[c, d]$ is the pair $[1, 0]$, there are rows r_1 and r_2 in $(A[K] - x)[c, d]$ containing pairs $[0, 1]$ and $[1, 1]$, respectively. Since $e(c) \preceq e(c')$, the row r_1 in $(A[K] - x)[c, c', d]$ have triple $[0, 0, 1]$. If the triple in the row r_2 is $[1, 1, 1]$, c' and d conflict and we are done. Otherwise, the row r_2 in $A[c']$ contains 0. Since, K is connected the column c' in $A[K] - x$ must contain 1 in some row r_3. Since $e(c) \preceq e(c')$, row r_3 in $A[c]$ contains 1. Since $A[K] - x$ has no conflict, the triple in the row r_3 in $A[c, c', d]$ is $[1, 1, 1]$. Now, x and the rows r_1 and r_3 give a conflict between c' and d, cf. Figure 3.3(b). Since, edges in $\{e(c)\}_{L_1}$ lie on one directed path, the corresponding sets $\{N(c)\}_{c \in L_1}$ form a chain. $\square$

Since bi-inclusive graphs are chordal bipartite graphs, a minimum vertex cover of a bi-inclusive graph can be found in linear time given some additional information on the graph [27]. Hence we have the following.

Observation 3. *A minimum vertex cover in a bi-inclusive graph can be found in $O(m \log m)$ time and in linear time ($O(m)$) if the chain order of vertices in one partition is given.*

Proof. Let L and R be the two bipartitions of a bi-inclusive graph. Let $x_1, \dots, x_\ell$ be the order of vertices in L such that $N(x_1) \supseteq N(x_2) \supseteq \cdots \supseteq N(x_\ell)$ and $y_1, \dots, y_r$ be the order of vertices in R such that $N(y_1) \supseteq N(y_2) \supseteq \cdots \supseteq N(y_r)$. Let $S \subseteq L \cup R$ be a minimum vertex cover. We observe that if $x_i \in S$, then also $x_j \in S$ for every $j < i$. Indeed, $N(x_i) \subseteq N(x_j)$ and if $x_j \notin S$, then all vertices in $N(x_j)$ must be in S. Obviously, after removing x_i from S we would obtain a smaller vertex cover, which would be a contradiction. Hence, $S \cap L = \{x_1, \dots, x_i\}$ for some $i \leq \ell$, and similarly, $S \cap R = \{y_1, \dots, y_j\}$ for some $j \leq r$. Moreover, there is no edge between $L \setminus S$ and $R \setminus S$. It is enough to find for every i, the smallest $j(i)$ such that $(x_{i+1}, y_{j(i)+1})$ is not an edge. The minimum cover are the sets L_i and $R_{j(i)}$ with the minimum sum $i + j(i)$. Obviously, $j(i) = \deg(x_{i+1})$.

Assuming we know the order and degree of vertices in one bipartition of the bi-inclusive graph, one can find the minimum vertex cover in linear time. If we know only degrees, to find the order of vertices, it is enough to sort vertices in one partition by their degree. □

The following example shows that bi-inclusiveness is not sufficient to guarantee existence of a gall.

Example 3. Consider the following matrix A and its conflict graph

$$\begin{array}{c|ccc} & c_1 & c_2 & c_3 \\ \hline r_1 & 1 & 0 & 0 \\ r_2 & 1 & 1 & 0 \\ r_3 & 1 & 1 & 1 \\ r_4 & 0 & 1 & 1 \\ r_5 & 0 & 1 & 0 \end{array}$$

Obviously, the conflict graph of A is bi-inclusive. However, there is no sequence x such that $A - x$ has a perfect phylogeny. On the other hand, if we delete row r_2 from A, the conflict graph of the new matrix A' remains the same, but there is a sequence x (either $x = 100$ or $x = 111$) such that $A' - x$ has a perfect phylogeny. We can learn from this example that the conflict graph does not contain enough information to decide whether there is a galled-tree network explaining a given matrix.

Chapter 4

A Polynomial Algorithm for One Special Case of the Galled-Tree Network Haplotyping Problem

The problem of determining haplotypes from genotypes has gained considerable prominence in the research community. Present algorithmic tools for haplotyping make effective use of phylogenetic trees ([48, 8, 16, 30, 24]). Based on some experimental results ([85, 60]), they assume that the evolutionary history for the original haplotypes, that form the current genotypes, satisfies the perfect phylogeny model. However these results do not fully exclude recombinations and models are needed that incorporate this extra degree of complication. Recently, Gusfield studied the two cases: haplotyping via imperfect phylogenies with a single homoplasy and via galled-tree networks with one gall ([94]). In Chapter 3 we characterized the existence of the galled-tree networks. Building on this, we present a polynomial algorithm for haplotyping via galled-tree networks with simple galls (having two mutations) ([44]). In the end, we give the experimental results comparing our algorithm with PHASE on simulated data.

4.1 Inferring Haplotypes via Galled-Tree Network

In Chapter 2 we introduced both the perfect phylogeny haplotyping (PPH) problem and galled-tree network haplotyping (GTNH) problem.

Given a genotype matrix A, for every $x, y \in \{0, 1\}$, recall 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 $[x, y]$, $[2, y]$ or $[x, 2]$. 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. Note that the conflict graph of every haplotype matrix inferred from A contains all edges of the conflict graph of A.

The next theorem by Bafna et al. characterizes the perfect phylogeny haplotyping problem. We give some notations before stating the theorem.

Definition 19. Given a genotype $n \times m$ matrix A, find a $2n \times m$ haplotype matrix B that is inferred from A. Let $\mathcal{X}_A = \{x_{r\{c_1,c_2\}};\ A(r,c_1) = A(r,c_2) = 2\}$ be a 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 $i \in \{1,\ldots,n\}$).

Theorem 14. *(Bafna et al.[8]) Given a genotype matrix A, there is a solution to PPH problem on A if and only if no two columns c_1, c_2 induce all three pairs $[1,1]$, $[0,1]$ and $[1,0]$ in A, and there exists an assignment I_B such that*

(1) for every $x_{rc_1c_2}, x_{r'c_1c_2} \in \mathcal{X}_A$, $I_B(x_{rc_1c_2}) = I_B(x_{r'c_1c_2})$;

(2) for every $x_{rc_1c_2} \in \mathcal{X}_A$, $I_B(x_{rc_1c_2}) = 0$ if c_1, c_2 induce $[1,1]$ in A;

(3) for every $x_{rc_1c_2} \in \mathcal{X}_A$, $I_B(x_{rc_1c_2}) = 1$ if c_1, c_2 induce both $[0,1]$ and $[1,0]$ in A; and

(4) for every $x_{rc_1c_2}, x_{rc_1c_3}, x_{rc_2c_3} \in \mathcal{X}_A$,
$$I_B(x_{rc_1c_2}) + I_B(x_{rc_1c_3}) + I_B(x_{rc_2c_3}) = 0.$$

When considering the GTNH problem, the above rules (except for (4)) do not apply since violating rules (1)–(3) just creates a conflict between corresponding columns/characters in B. Based on the above characterization by Bafna [8], we have the following lemma about the relationship between the conflict graph of B and the assignment I_B.

Lemma 15. *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:*

(C1) c_1, c_2 induce all three pairs: $[1,1]$, $[0,1]$ and $[1,0]$ in A;

(C2) $I_B(x_{rc_1c_2}) \neq I_B(x_{r'c_1c_2})$ *for some* $x_{rc_1c_2}, x_{r'c_1c_2} \in \mathcal{X}_A$;

(C3) c_1, c_2 *induce* $[1,1]$ *in* A, *and there is* $x_{rc_1c_2} \in \mathcal{X}_A$ *such that* $I_B(x_{rc_1c_2}) = 1$;

(C4) c_1, c_2 *induce* $[0,1]$ *and* $[1,0]$ *in* A, *and there is* $x_{rc_1c_2} \in \mathcal{X}_A$ *with* $I_B(x_{rc_1c_2}) = 0$.

4.2 A Special Instance of GTNH Problem

The GTNH problem seems to be very hard, hence we consider a special instance of this problem. We restrict the problem on genotype matrices which satisfy a structural condition called the **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 a_{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* $[1,1]$.

Let us state two observations about the weak property. First, a genotype matrix with the weak property 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 [20] and most of the simulation data sets which we randomly generated genotypes from Hudson's simulation program [61] satisfy the weak property.

We further simplify the problem by requiring that the solution is a galled-tree network with galls containing two mutation edges. We call such networks *simple GTN* and the problem of deciding whether a genotype matrix can be explained by a simple GTN, the SGTNH problem. We will show that the SGTNH problem is polynomial for matrices with the weak property.

Theorem 3 gives a simple characterization of haplotype matrices which can be explained by a simple galled-tree network. In particular, we have the following observation.

Observation 4. *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). Furthermore, given a genotype matrix A, if A can be explained by a simple galled-tree network then the conflict graph of A contains only isolated edges.*

Hence, from now on we assume that the conflict graph of A has only isolated edges (otherwise, the algorithm reports *fail*). Let us call such a matrix *simple conflicts genotype matrix.* A haplotype matrix B explainable by an SGTN inferred from a genotype matrix A will be called a *solution.*

4.3 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.

4.3.1 Phase 1: Eliminating Rows with Less than Three or More than Four 2's.

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 $I_R(x_{rc_1c_2}) = a_{ij}$).*
2. *For every row r with 2's in columns $c_1, c_2, \ldots, c_k$, where $k \geq 5$:*
   (a) *For every 5-tuple $c_i, \ldots, c_{i+4}$, where $i = 1, \ldots, k-4$, replace row r in $A[c_i, \ldots, c_{i+4}]$ with all (16) possible inferrings of $r[c_i, \ldots, c_{i+4}]$ to obtain 16 matrices $A^i_1, \ldots, A^i_{16}$.*
   (b) *Find the matrix $A^i_{j_i}$ whose conflict graph has only isolated edges. It can be proved that 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.

In order to prove the correctness of Procedure 1, we have the following claims, and their proofs respectively. They state that rows in A with one, two or more than four 2's can be inferred to A' such that A' has SGTNH solution if and only if A has.

Claim 9. *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 10. *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 $I_R(x_{rc_1c_2}) = a_{12}$. 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 $I_B = I_{B'} \cup \{(x_{rc_1c_2}, a_{12})\}$. 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 Observation 4, 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 Observation 4, 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 11. *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 $C = \{c_1, \ldots, c_5\}$. 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 the 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 Observation 4, 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 Observation 15, 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. □

4.3.2 Phase 2: Eliminating of Some Triples of 2's.

Now, 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. 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. Note that it can be proved that this inferring is unique.*

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.*

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

The following claims and their proofs show the correctness of Procedure 2. In particular, under certain circumstances, rows with three or four 2's in A has unique solution and therefore can be preprocessed.

Claim 12. *Given a simple conflicts 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, $a_{12} \oplus a_{23} \oplus a_{13} = 0$ 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, $a_{12} \oplus a_{23} \oplus a_{13} = 1$ 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 a simple conflicts genotype matrix. 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 $a_{ij} \oplus I_B(x_{rc_ic_j}) = 1$, c_i and c_j conflict in B. Observe that

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

First, assume that $a_{12} \oplus a_{23} \oplus a_{13} = 0$. If B is a solution then $a_{12} \oplus I_B(x_{rc_1c_2}) = a_{23} \oplus I_B(x_{rc_2c_3}) = a_{13} \oplus I_B(x_{rc_1c_3}) = 0$, 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 $I_B(x_{rc_1c_2}) = a_{12}$, $I_B(x_{rc_2c_3}) = a_{23}$ and $I_B(x_{rc_1c_3}) = a_{13}$, 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 10).

Second, assume that $a_{12} \oplus a_{23} \oplus a_{13} = 1$. If B is a solution then exactly one of $a_{12} \oplus I_B(x_{rc_1c_2})$, $a_{23} \oplus I_B(x_{rc_2c_3})$ and $a_{13} \oplus I_B(x_{rc_1c_3})$ 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 13. *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 a simple conflicts genotype matrix. Second, assume that it contains at least one isolated edge, say c_1, c_2. Then, by Claim 12, 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 12, 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 12, 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. □

4.3.3 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 20 (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 4 (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.

We have the following characterization of solutions to the SGTNH problem.

Lemma 16. *Let A be a simple conflicts 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 Observation 4 every non-trivial component of the conflict graph G_B has 2 vertices, i.e., is a single edge. By Claim 12(3) and Claim 13 (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. □

4.3.4 Solving Hypergraph Covering Problem

Now, it suffices to show that the HC problem can be solved in polynomial time. The following lemma describes the easiest case which can be solved by first identifying and covering cycles, and then greedily covering the remaining 3-edges.

Lemma 17. *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 $O(n^2)$.*

We start with the following definition.

Definition 21. Let H be a hypergraph. A *3-edge-path* of length k is a subhypergraph of H consisting of k 3-edges $e_1 = \{v_1, v_2, v_3\}$, $e_2 = \{v_3, v_4, v_5\}$, ..., $e_k = \{v_{2k-1}, v_{2k}, v_{2k+1}\}$, 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 4.1.

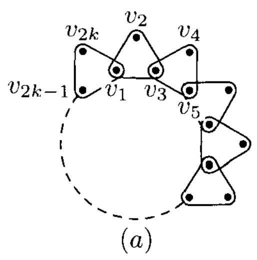


Figure 4.1: Example of 3-edge-cycle.

Observation 5. *If the input hypergraph for the GI 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 is odd*$\}$ *and the other of edges* $\{(v_i, v_{i+1})$; *i is even*$\}$, *where* $v_{2k+1} = v_1$. *In any solution, all vertices are incident to an edge of the solution.*

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 6. *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 5, 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 e, there is at most one choice for an edge of e 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.

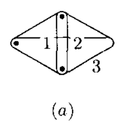
Now, assume there is a 3-edge left in H. By applying Observation 6 in each iteration of the above process, it is easy to see that all edges in G (with exception of the edges from C) are necessarily in the solution of the HC problem for H. If there is a 3-edge in H with at least two vertices incident 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. □

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 7. *Let H be a hypergraph with only 3-edges and let C be 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 4.2(a). The covering for the 3-edge-cycle is equivalent to the covering for the 4-edge formed by the four vertices of the three 3-edges. Figure 4.2(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. We need the definition of a special structure before stating the lemma.

Definition 22. Let H be a hypergraph. A *3-edge-chain* of length k is a subhypergraph of H composed of k 3-edges $e_1 = \{v_1, v_2, v_3\}$, $e_2 = \{v_2, v_3, v_4\}$, $\ldots$, $e_k = \{v_k, v_{k+1}, v_{k+2}\}$, where vertices $v_1, \ldots, v_{k+2}$ are all distinct, cf. Figure 4.3. The vertices v_1 and v_{k+2} are the *end vertices* of the chain.



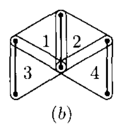


Figure 4.2: (a) Doubly incident 3-edge-cycle; (b) Forcing configuration of four doubly connected 3-edges.

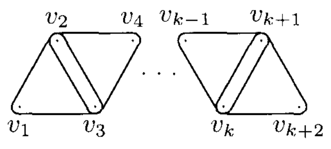


Figure 4.3: Example of a 3-edge-chain of length k.

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

Observation 8. *If the input hypergraph for the HC problem is a 3-edge-chain* $e_1 = \{v_1, v_2, v_3\}$, $e_2 = \{v_2, v_3, v_4\}$, $\ldots$, $e_k = \{v_k, v_{k+1}, v_{k+2}\}$, *where* $k \geq 3$ *then there are exactly two solutions. One solution consists of edges* $\{(v_i, v_{i+1});$ *i is odd*$\}$ *and the other of edges* $\{(v_i, v_{i+1});$ *i 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.*

Note that if in Definition 22, $v_1 = v_{k+2}$, $v_2 = v_{k+1}$ and all other vertices are distinct, then this is a special 3-edge-chain with every 3-edge doubly incident on two other 3-edges. Based on Observation 8, there is unique covering for such chain with even number of 3-edges, and no covering for such chain with odd number of 3-edges.

Lemma 18. *Let C be a doubly connected component of a hypergraph H. Let C contain only 3-edges and no doubly incident 3-edge-cycles such that every pair of vertices belongs to at most two 3-edges. By covering and remove the 3-edges of C that have unique solutions, the remaining 3-edges of C form a 3-edge-chain.*

Proof. First, if C has at most two 3-edges, it is a 3-edge-chain of length 2, cf. Figure 4.3.

Second, consider C contains three 3-edges e_1, e_2 and e_3. There are two possibilities how C can look like. Without loss of generality, assume e_1 and e_2 are doubly incident. If e_3 is doubly incident with either one of e_1 or e_2, C forms a 3-edge-chain. If e_3 is doubly incident on both e_1 and e_2, we get a doubly incident 3-edge-cycle, a contradiction.

Now, consider C contains four 3-edges. There are only four possibilities how C can look like, depicted in Figure 4.4. In the first case, we have a 3-edge chain. In the second case, by Observation 7, there is either no covering or a unique covering of C. In the third

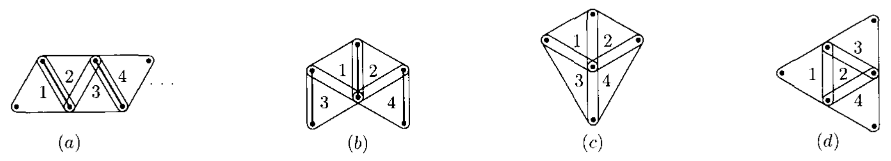


Figure 4.4: All possible cases for a doubly connected component with four 3-edges.

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. Furthermore, assume we add a new 3-edge e to C to get C'. e will doubly incident on at least one 3-edge in C. According to Observation 6 and Observation 8, the covering for e is either uniquely determined (with 0 or 1 covering) or e together with C form a new 3-edge-chain. It follows that every doubly connected component, after removing the 3-edges that can be uniquely covered, is a 3-edge chain. □

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

Theorem 19. *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 20 are satisfied, or

(T2) the choice for pairs of vertices from this hyperedge is postponed and binded to the 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 the 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 17 on the modified H (containing only unprocessed edges). Union of G and a solution from the algorithm of Lemma 17 gives a solution to the HC problem.

The set of rules:

1. If there is an hyperedge e contained in another hyperedge f, remove e from H. As the covering for f contains the covering for e, this will not affect the solution to the problem. We proceed as in the case (T2). 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 4.5.

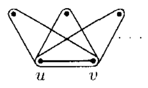


Figure 4.5: The pair u, w contained in at least three 3-edges.

3. If H contains doubly incident 3-edge-cycle, cf. Figure 4.2(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 4.6 for all possible cases. 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.

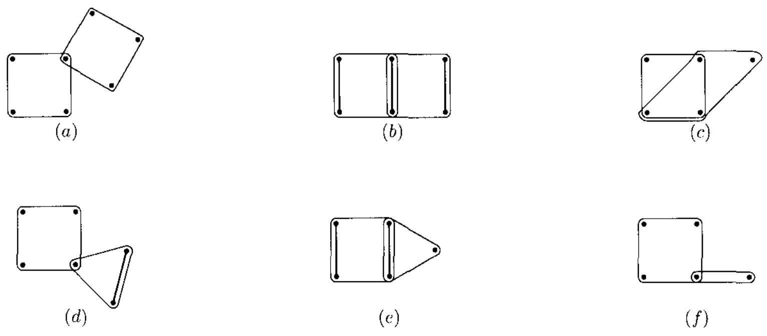


Figure 4.6: An 4-edge intersecting other hyperedges. The thick edges show the covering if it exists. In cases (a), (c) and (f) there is no solution.

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 18, 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. 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.

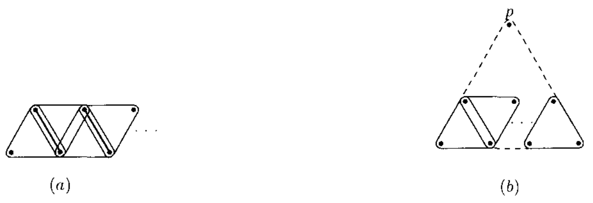


Figure 4.7: (a) An example of smaller solution to a 3-edge-chain of even length. (b) Replacement of a 3-edge-chain of odd length by a new 3-edge.

If a component C is a 3-edge-chain of even length,there is a solution s for C which does not contain the end vertices, cf. Figure 4.7(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,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 p, cf. Figure 4.7(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, p) is picked, cover the 3-edge-chain by the solution that contains u; and similarly for the edge (v, p).

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 17 takes time $O(n^2)$ and finally, resolution of binded hyperedges can be done in $O(n)$ time. □

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.

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 $O(n^2+nm^2)$.

4.3.5 Experimental Results

We implemented the algorithm and performed the following experiments on simulation data. We compared our algorithm's performance with PHASE v2 [103, 102]. The results show that our algorithm seems promising for the corresponding haplotyping problem.

In particular, the data is prepared in three steps. First, binary matrices, each having a perfect phylogenetic tree, are generated using Hudson's [61] simulation program with recombination rate being 0. Then, for each matrix's perfect phylogenetic 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 based on three standards [103]: 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* ([73]) 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 four sets of 100 matrices. The experimental results are shown in Table 4.1.

The results show that our algorithm has comparable accuracy with PHASE while runs tens to hundreds times faster than the statistical method.

Table 4.1: Comparison of accuracy between our algorithm (SGTN) and PHASE on the four 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 duplicated columns. Each rate is normalized by the number of genotype matrices in that set.

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

Chapter 5

Galled-Tree Network Haplotyping Problem is NP-Complete

In this chapter, we study another special case of the GTNH problem ([42]). From this version of the problem, we are able to show the NP-completeness of the GTNH problem ([43]).

5.1 A Special Instance of GTNH Problem

For the instance studied in this chapter, we put no restriction on the galled-tree networks but assume a little bit stronger assumption about the input genotype matrices.

Definition 23 (Weak diagonal property). Given a genotype matrix A, we say that a pair of SNPs is *active* if it contains $[2,2]$, or it induces all three pairs $[1,1]$, $[0,1]$ and $[1,0]$. Further, we say that a pair c_1, c_2 is *weakly active* if either it is active, or if there is a SNP c_3 such that c_1, c_3 and c_2, c_3 are both active pairs. We say that A has the *weak diagonal (WD) property* if every weakly active pair of SNPs induces both $[0,1]$ and $[1,0]$.

Our experiments show that many data sets, including real data and simulated data sets, satisfy the WD property. We tested Daly's genotype data ([20]), which has 103 columns and 387 rows. The data contains 11 blocks, where the evolutionary history for haplotypes in each block is believed to have very few recombinations. Such data is in particular suitable for haplotyping via GTN as there is a small chance of interaction among a small number of recombination cycles. The genotype matrices for 9 out of the 11 blocks satisfy WD property. The genotype matrices for the remaining two blocks have the WD property after removal of only one column. We also tested the WD property on a real data set (genotypes_chr1_JPT_CHB_r21_nr_fwd_phased) downloaded from the international HapMap project website ([106]). The data set is a phased haplotype data for different individuals, with every two adjacent haplotypes coming from one individual. We read the first 300 rows of the data, with each row having 200 SNP values and generated the corresponding genotype data by joining the adjacent haplotypes. In order to test the WD property for each block of genotypes, where recombinations are assumed to be rare, we assumed that each block has size 5. This is consistent with the findings in [38] showing that the majority of blocks found in human genome have a small size ($<$ 5 kbp, i.e., approximately, $<$ 5 SNPs, as the data has a SNP in roughly every 1 kbp). We moved a window of size 5 through the data to get the hypothetical blocks. The percentage of these matrices that satisfy WD property increases as the frequency of their minor alleles increases (see Table 5.1). When the minimum minor allele frequency is at least 30%, all the matrices have the WD property.

minimum minor allele frequency	$\geq 0.1\%$	$\geq 1\%$	$\geq 5\%$	$\geq 10\%$	$\geq 20\%$	$\geq 30\%$
percent of WD matrices	5.6%	5.9%	11.3%	18.4%	30.8%	100%

Table 5.1: The percentage of matrices that have the WD property increases as the minimum minor allele frequency of the matrices increases.

If A has the WD property, then it follows from Lemma 15 that for every $x_{rc_1c_2} \in \mathcal{X}_A$ such that $I_B(x_{rc_1c_2}) = 0$, c_1 and c_2 conflict in B. Reformulate the lemma for matrices with the WD property, we get the following observation.

Observation 9. *Given a genotype matrix A with the WD property, let B be a haplotype matrix inferred from A. SNPs $c_1 < c_2$ conflict in G_B if and only if either they induce $[0,1]$, $[1,0]$ and $[1,1]$ in A, or there is $x_{rc_1c_2} \in \mathcal{X}_A$ such that $I_B(x_{rc_1c_2}) = 0$. Consequently, if a pair $c_1 < c_2$ conflicts in B, then it is active in A.*

First, we will study some basic characteristics of matrices with the WD property. In the following subsections, we will use these results to reduce the problem to a hypergraph covering problem similar to the one introduced in Chapter 4([44]).

5.1.1 Basic Properties of Matrices with the WD Property

In this subsection we observe some properties of matrices with the WD property which can be explained by a GTN, in particular, we show that it is sufficient to study genotype matrices with moderate number of 2's per row.

Claim 14. *Given an $n \times m$ genotype matrix A, assume that A has a row r which contains one 2. Let A' be a 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. The claim follows by Observation 1. $\square$

Claim 15. *Given a genotype matrix A with the WD property, let B be a matrix inferred from A such that it can be explained by a GTN. Then for every triple of 2's occurring in columns $c_1 < c_2 < c_3$ and in the same row r, exactly one of $I_B(x_{rc_1c_2}), I_B(x_{rc_1c_3}), I_B(x_{rc_2c_3})$ is equal to 0.*

Proof. The values have to satisfy condition (4) of Theorem 14, i.e., $I_B(x_{rc_1c_2})+I_B(x_{rc_1c_3})+I_B(x_{rc_2c_3}) = 0$. This implies that either all three variables are mapped to 0 or exactly one of them. In the latter case we are done. In the former case, due to the WD property, by Observation 9, we have that all three pairs c_1, c_2, c_1, c_3 and c_2, c_3 conflict in B. Hence, the conflict graph G_B is not bipartite, a contradiction with Theorem 3. $\square$

The following two corollaries easily follow from the above claim.

Corollary 4. *Given a genotype matrix A with the WD property, let B be a matrix inferred from A such that it can be explained by a GTN. Then for every four 2's occurring in columns $c_1 < c_2 < c_3 < c_4$ and in the same row r, there are pairs $d_1 < d_2$ and $d_3 < d_4$ such that $\{d_1, d_2, d_3, d_4\} = \{c_1, c_2, c_3, c_4\}$, $I_B(x_{rd_1d_2}) = I_B(x_{rd_3d_4}) = 0$ and for every pair $d < d'$, where $d, d' \in \{c_1, c_2, c_3, c_4\}$, different from $d_1 < d_2$ and $d_3 < d_4$, $I_B(x_{rdd'}) = 1$.*

Corollary 5. *Given a genotype matrix A with the WD property, if A has a row r with at least five 2's, then A cannot be explained by a GTN.*

Proof. Let $C_r = \{c_1, \dots, c_\ell\}$ be the ordered set of all columns containing 2 in the row r. Suppose $\ell \geq 5$. Assume that A can be explained by a GTN, and let B be the corresponding haplotype matrix. By Claim 15, for every three SNP $c_i < c_j < c_k \in C_r$ exactly one of $I_B(x_{rc_1c_2}), I_B(x_{rc_1c_3}), I_B(x_{rc_2c_3})$ is equal to 0. Without loss of generality we can assume that for a triple c_1, c_2, c_3, we have $I_B(x_{rc_1c_2}) = 0$. It follows that the values $I_B(x_{rc_1c_3})$, $I_B(x_{rc_2c_3})$, $I_B(x_{rc_1c_4})$, $I_B(x_{rc_2c_4})$, $I_B(x_{rc_1c_5})$, $I_B(x_{rc_2c_5})$ are all equal to 1, and hence the values $I_B(x_{rc_3c_4})$, $I_B(x_{rc_3c_5})$, $I_B(x_{rc_4c_5})$ are all equal to 0, a contradiction. $\square$

Based on Claim 14 and Corollary 5, we have the following interesting corollary.

Corollary 6. *Given a genotype matrix A with the WD property, if each row contains less than two or more than four 2's, then the GTNH problem for A can be solved in polynomial time.*

From now on, we will assume that the input genotype matrix (with the WD property) has either zero, two, three or four 2's in each row, since otherwise either it cannot be explained by a GTN (Corollary 5) or it can be converted to another matrix with the WD property without affecting existence of a solution (Claim 14).

5.1.2 Genotype Hypergraph and Its Coverings

In Chapter 4 we gave the definition of genotype-hypergraph. Here, we gave a similar definition for matrices that have WD property. In particular, we define a hypergraph assigned to a genotype matrix and its coverings, and observe its connection to the GTNH problem.

Definition 24 (Genotype hypergraph and its coverings)**.** Given an $n \times m$ genotype matrix A with the WD property, the *genotype hypergraph* H_A of A has the set of SNPs $\{1, \ldots, m\}$ as a vertex set, and for every row r of A containing at least two 2's, say in columns $c_1, \ldots, c_k$, there is an *unforced* hyperedge $e_r = \{c_1, \ldots, c_k\}$. Furthermore, for every two columns c and c' inducing $[0,1]$, $[1,0]$ and $[1,1]$ in A, there is a *forced* hyperedge $[c, c']$ in H_A. The hypergraph H_A does not contain any other hyperedges. Since for any $k > 2$, there is no forced k-edge, we will omit the word unforced when referring to such a hyperedge.

Consider a hypergraph H_A with hyperedges of cardinality at most 4. We say that a graph G on the vertex set $V(H_A)$ *covers* H_A if G can be obtained as follows:

- for every forced 2-edge $[c_1, c_2]$ of H_A, add the edge (c_1, c_2) in G;
- for every unforced 2-edge $\{c_1, c_2\}$ of H_A, make a choice whether to add the edge (c_1, c_2) in 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.

A graph G that covers H_A is called a *covering* of H_A.

Now, we can characterize all possible conflict graphs of matrices inferred from an input genotype matrix as follows.

Lemma 20. *Given a genotype matrix A with the WD property, let B be a haplotype matrix inferred from A which can be explained by a GTN. Then the conflict graph G_B of B is a covering of the genotype hypergraph H_A of A.*

Conversely, let G be a covering for the graph H_A. Each way of finding the covering G from H_A (a collection of choices for every unforced hyperedge of H_A), defines a haplotype matrix B. This haplotype matrix B can be inferred from A and the conflict graph of B is G.

Proof. Let B be a haplotype matrix inferred from A which can be explained by a GTN. By Corollary 5, A contains at most four 2's in every row. By Observation 9, SNPs $c_1 < c_2$ conflict if and only if they induce $[0, 1]$, $[1, 0]$ and $[1, 1]$ in A, or if $I_B(x_{rc_1c_2}) = 0$ for some $x_{rc_1c_2} \in \mathcal{X}_A$. It follows by Definition 20, Claim 15 and Corollary 4 that G_B covers H_A.

Conversely, let G be a covering for H_A. Then A contains at most four 2's in every row, and there exists a choice for every (unforced) k-edge as described in Definition 20. By Claim 15 and Corollary 4, each such collection of choices for every unforced k-edge, defines values $I_B(x_{rcc'})$ for every $x_{rcc'} \in \mathcal{X}_A$ satisfying condition (4) of Theorem 14. Hence, also a haplotype matrix B can be inferred from A. By Observation 9, the conflict graph G_B is isomorphic to G. □

5.1.3 Characterization of Conflict Graphs of Haplotype Matrices Explainable by a GTN Inferred from a Genotype Matrix with the WD Property

The following claim is crucial in restricting the possible cases we need to study.

Claim 16. *Given a genotype matrix A with the WD property, let B be a matrix inferred from A which can be explained by a GTN. Let c_1, c_2, c_3 be three SNPs such that both pairs c_1, c_2 and c_2, c_3 conflict in B. Let K be the component containing c_1, c_2, c_3 and x a vector such that $B[K] - x$ has no conflicts. Then $x[c_1, c_2, c_3]$ is either $[0, 1, 1]$ or $[1, 1, 0]$.*

Proof. By Observation 9, both pairs c_1, c_2 and c_2, c_3 are active. Hence, the pair c_1, c_3 is weakly active and satisfies the weak diagonal condition, i.e., it induces $[0, 1]$ and $[1, 0]$ in A. Hence, there are two rows in B containing those two pairs in $B[c_1, c_3]$. Note that c_1 and c_3 cannot conflict in B, otherwise G_B is not a bipartite which would violate (1) of Theorem 3. Therefore, since x is a sequence in one of the rows of $B[K]$, we have $x[c_1, c_3] \neq [1, 1]$. On the other hand x has to remove conflicts between c_1, c_2 and between c_2, c_3, i.e, $x[c_1, c_2], x[c_2, c_3] \neq [0, 0]$. There are only three possibilities left for the value of $x[c_1, c_2, c_3]$: $[0, 1, 0]$, $[0, 1, 1]$ and $[1, 1, 0]$.

To finish the proof it is enough to consider (and exclude) the case $x[c_1, c_2, c_3] = [0, 1, 0]$. Since, c_1, c_2 conflict in B, there is a row r containing triple $[1, 1, y]$, where $y \in \{0, 1\}$, in $B[c_1, c_2, c_3]$. If $y = 1$, then c_1 and c_3 conflict in B, a contradiction. On the other hand, if $y = 0$, then $B[K] - x$ still contains the row r. Thus, SNP c_2 and c_3 still conflict in $B[K] - x$, a contradiction. □

Claim 16 is a powerful tool which helps to characterize all possible conflict graphs of haplotype matrices explainable by a GTN inferred from a given genotype matrix with the WD property.

Corollary 7. *Given a genotype matrix A with the WD property, let B be a matrix inferred from A which can be explained by a GTN. Every vertex in G_B has degree at most 2, i.e., G_B consists of cycles and paths.*

Proof. Assume to the contrary that there is an SNP c with a degree at least 3 in the conflict graph G_B. Let c_1, c_2, c_3 be three neighbors of c in G_B. Let K be the component containing c, c_1, c_2, c_3 and x a sequence such that $B[K] - x$ does not contain any conflict. By Claim 16, $x[c_1, c, c_2]$ is either $[0, 1, 1]$ or $[1, 1, 0]$. Without loss of generality assume it is $[0, 1, 1]$. By Claim 16, $x[c_1, c, c_3] = [0, 1, 1]$. Now, $x[c_2, c, c_3] = [1, 1, 1]$, which contradicts Claim 16. □

As a consequence we have the following characterization.

Lemma 21. *Given a genotype matrix A with the WD property, let B be a matrix inferred from A which can be explained by a GTN. Then each component of G_B is a path of length at most 3 (contains at most three edges).*

Proof. By Corollary 7, each component of G_B is either a cycle or a path. Assume to the contrary that a component K of G_B is a cycle or a path of length at least 4. Since by Theorem 3, K is a bipartite, if K is a cycle then its length is at least 4. Hence, K contains a path $(c_1, c_2, c_3, c_4, c_5)$, where SNPs c_1 and c_5 can be the same (in the case when K is a cycle of length 4). Let x be a sequence such that $B[K] - x$ does not contain any conflict. By Claim 16, each of $x[c_1, c_2, c_3]$, $x[c_2, c_3, c_4]$ and $x[c_3, c_4, c_5]$ is either $[0, 1, 1]$ or $[1, 1, 0]$. Obviously, this is not possible, a contradiction. □

5.1.4 Dealing with Rows with Two 2's and Canonical Coverings

In this section we deal with the problem how to resolve rows containing only two 2's. The general strategy is to resolve such rows unequally if it helps to avoid conflict, and otherwise equally. The following two claims show the correctness of this strategy.

Claim 17. *Given a genotype matrix A with the WD property, consider any two columns $c_1 < c_2$. Let $r_1, \ldots, r_k$ be rows containing 2's in columns c_1, c_2 and no other 2's, and*

$r'_1, \dots, r'_\ell$ rows containing 2's in columns c_1, c_2 and at least one 2 in some other column. Assume that $k \geq 1$ and that c_1 and c_2 do not induce $[1,1]$ in A. Let B be a haplotype matrix inferred from A such that

- *B can be explained by a GTN;*
- *for some $i = 1, \dots, k$, $I_B(x_{r_i c_1 c_2}) = 0$; and*
- *for every $i = 1, \dots, \ell$, $I_B(x_{r'_i c_1 c_2}) = 1$.*

Then the matrix B' such that for every $i = 1, \dots, k$, $I_{B'}(x_{r_i c_1 c_2}) = 1$ and for every other $x_{rcc'} \in \mathcal{X}_A$, $I_{B'}(x_{rcc'}) = I_B(x_{rcc'})$, can be explained by a GTN as well.

Proof. The conflict graph $G_{B'}$ differs from the conflict graph G_B only by not containing an edge (c_1, c_2). Hence, it satisfies condition (1) of Theorem 3. Let us verify the second condition. Consider component of G_B that contains c_1 and c_2. By Lemma 21, it is a path of length at most 3. Therefore, in $G_{B'}$ this component is split into two, each containing one of c_1, c_2. Consider a component K of $G_{B'}$. Since it contains at most one of c_1 and c_2 then $B[K]$ and $B'[K]$ contains the same set of sequences. Let K' be a component of G_B containing K. There is a sequence x such that $B[K'] - x$ has no conflict. Obviously, $B[K] - x[K]$ has no conflict as well, and hence also $B'[K] - x[K]$. By Theorem 3, B' can be explained by a GTN.

□

Claim 18. *Given a genotype matrix A with the WD property, consider any two columns $c_1 < c_2$. Let $r_1, \dots, r_k$ be rows containing 2's in columns c_1, c_2 and no other 2's, and $r'_1, \dots, r'_\ell$ rows containing 2's in columns c_1, c_2 and at least one 2 in some other column. Assume that $k \geq 1$. Let B be a haplotype matrix inferred from A such that B can be explained by a GTN. If either for some $i = 1, \dots, \ell$, $I_B(x_{r'_i c_1 c_2}) = 0$ or c_1 and c_2 induce $[1,1]$ in A, then the matrix B' such that for every $i = 1, \dots, k$, $I_{B'}(x_{r_i c_1 c_2}) = 0$ and for every other $x_{rcc'} \in \mathcal{X}_A$, $I_{B'}(x_{rcc'}) = I_B(x_{rcc'})$, can be explained by a GTN as well.*

Proof. The conflict graph $G_{B'}$ is the same as the conflict graph G_B. Hence, it satisfies the first condition of Theorem 3. It is enough to verify the second condition. Consider a component K of $G_{B'} = G_B$. If it does not contain c_1 and c_2 then $B[K]$ and $B'[K]$ contains the same set of sequences, and condition (2) easily follows.

Since c_1 and c_2 conflict, we only need to consider the case when K contains both c_1 and c_2. If K does not contain any other SNP, condition (2) trivially holds. Without loss of generality, assume that K contains an SNP c_3 such that c_2, c_3 conflicts in B. Similarly, if for every $i = 1, \dots, k$, $I_B(x_{r_i c_1 c_2}) = 0$, then $B' = B$, and the claim follows trivially.

Hence, we can assume that there is $i \in \{1, \ldots, k\}$ such that $I_B(x_{r_i c_1 c_2}) = 1$. We will show that this is not possible. Since, B can be explained by a GTN, there is a sequence x such that $B[K] - x$ has no conflict. By Claim 16, $x[c_1, c_2, c_3]$ is either $[0, 1, 1]$ or $[1, 1, 0]$. If $A[c_1, c_2, c_3]$ contains $[2, 2, 0]$ in row r_i then $B[c_1, c_2, c_3]$ contains $[0, 1, 0]$ in row $2r_i - 1$ or row $2r_i$. Hence, if $x[c_1, c_2, c_3] = [0, 1, 1]$, x cannot remove the conflict between c_1 and c_2, and if $x[c_1, c_2, c_3] = [1, 1, 0]$, x cannot remove the conflict between c_2 and c_3. On the other hand, if $A[c_1, c_2, c_3]$ contains $[2, 2, 1]$, then one of the rows $2r_i - 1$ and $2r_i$ contains $[1, 1]$ in $B[c_1, c_3]$ and hence c_1 and c_3 conflict in B. This is a contradiction, since by Theorem 3, G_B is bipartite. □

Based on Claims 17 and 18, we have the following strategy how to deal with rows with only two 2's.

Lemma 22. *Given a genotype matrix A with the WD property, the matrix A can be explained by a GTN if and only if there is a matrix B which can be explained by a GTN and is inferred from A such that for every row r with only two 2's, say in columns c_1 and c_2,*

- *$I_B(x_{rc_1c_2}) = 0$, if either c_1, c_2 induce $[1, 1]$ in A or there is a row r' with at least three 2's such that $x_{r'c_1c_2} \in \mathcal{X}_A$ and $I_B(x_{r'c_1c_2}) = 0$;*
- *$I_B(x_{rc_1c_2}) = 1$, otherwise.*

In other words, a row with only two 2's is never resolved in a way which would introduce a conflict which is not introduced by another row with more than two 2's. This leads us to the following definition of a special type of hypergraph coverings.

Definition 25 (Canonical covering)**.** We say that a graph G is a *canonical covering* of a hypergraph H if for every unforced 2-edge $\{c_1, c_2\}$ of H, the edge (c_1, c_2) is formally added to G if and only if it is added to G by some other hyperedge than an unforced 2-edge $\{c_1, c_2\}$.

Note that although adding edges to G for some unforced 2-edges of H in a canonical covering does not affect the resulting graph covering, it affects the process of inferring which defines the matrix B. Using the above definition we can reformulate Lemma 22 as follows.

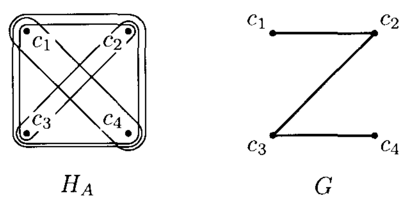
Corollary 8. *Given a genotype matrix A with the WD property, if A can be explained by a GTN then there is a haplotype matrix B inferred from A which can be explained by a GTN and its conflict graph is a canonical covering of H_A.*

5.1.5 Extended Hypergraph Covering Problem — Adding Switches

To convert the GTNH problem for input genotype matrices with the WD property to a hypergraph covering problem, we need to observe the last important conditions that have to be satisfied if the input genotype matrix is explainable by a GTN, and consequently, we will generalize the hypergraph covering problem by adding "switches". In the next subsection we then show that this generalized definition of the hypergraph covering problem is rich enough to characterize the GTNH problem for WD matrices.

It is easy to see that a conflict graph of a matrix inferred from a given genotype matrix A which can be explained by a GTN, is also a covering of hypergraph H_A which satisfies all the observed conditions. The following examples show that these conditions are not sufficient.

$$\begin{array}{c|cccc} A & c_1 & c_2 & c_3 & c_4 \\ \hline & 2 & 2 & 2 & 0 \\ & 0 & 2 & 2 & 2 \\ & 2 & 0 & 2 & 2 \\ & 2 & 2 & 0 & 2 \end{array}$$



$$\begin{array}{c|cccc} B & c_1 & c_2 & c_3 & c_4 \\ \hline & 0 & 0 & 1 & 0 \\ & 1 & 1 & 0 & 0 \\ & 0 & 1 & 1 & 0 \\ & 0 & 0 & 0 & 1 \\ & 0 & 0 & 1 & 1 \\ & 1 & 0 & 0 & 0 \\ & 1 & 1 & 0 & 0 \\ & 0 & 0 & 0 & 1 \end{array}$$

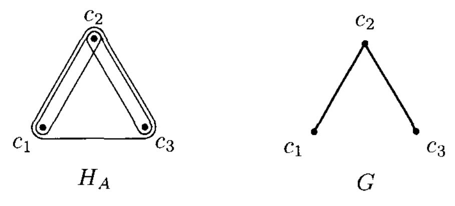
Figure 5.1: Example of a genotype matrix A with the WD property, the corresponding GI problem and its solution, and a haplotype matrix B inferred from A with the conflict graph equivalent to this solution. The matrix B does not satisfy condition (2) of Theorem 3.

Example 4. Figure 5.1 shows an example of a genotype matrix A with four columns (SNPs). It is easy to see that A has the WD property since every pair of columns induces $[0,1]$ and $[1,0]$. The corresponding hypergraph H_A has four 3-edges. One possible solution is a covering G of H_A which can be constructed as follows: in 3-edge $\{c_1, c_2, c_3\}$ we select edge (c_1, c_2), in 3-edge $\{c_2, c_3, c_4\}$ we select edge (c_2, c_3), in 3-edge $\{c_1, c_3, c_4\}$ we select edge (c_3, c_4) and finally, in 3-edge $\{c_1, c_2, c_4\}$ we select edge (c_1, c_2). Note that this covering contains only paths of length at most 3 and is canonical since A does not contain any rows with only two 2's. A haplotype matrix B corresponds to this selection of edges in G, hence its conflict graph is G. However, removal of any row from matrix B is not able to eliminate all conflicts in B. Hence, the condition (2) of Theorem 3 is not satisfied and B cannot be explained by a GTN.

Consider now another graph G' on the same set of SNPs with only two edges (c_1, c_4)

and (c_2, c_3). This graph is a covering for H_A as well. Obviously, the haplotype matrix corresponding to this selection of edges for every hyperedge of H_A satisfies condition (2) of Theorem 3, i.e., can be explained by a GTN. Hence, one of the additional constrains we will require from the new hypergraph covering problem is the minimality of the number of edges.

$$\begin{array}{c|ccc} A & c_1 & c_2 & c_3 \\ \hline & 2 & 2 & 2 \\ & 2 & 0 & 2 \\ & 0 & 1 & 1 \\ & 1 & 1 & 0 \\ & 0 & 1 & 0 \end{array}$$



$$\begin{array}{c|ccc} B & c_1 & c_2 & c_3 \\ \hline & 1 & 1 & 0 \\ & 0 & 0 & 1 \\ & 1 & 0 & 0 \\ & 0 & 0 & 1 \\ & 0 & 1 & 1 \\ & 1 & 1 & 0 \\ & 0 & 1 & 0 \end{array}$$

Figure 5.2: Example of a genotype matrix A with the WD property, the corresponding GI problem and its unique solution, and a haplotype matrix B inferred from A with the conflict graph equivalent to this solution. The matrix B does not satisfy condition (2) of Theorem 3.

Example 5. Figure 5.2 shows an example of a genotype matrix A with three SNPs which satisfies the WD property. It has a row with only two 2's, in columns c_1 and c_3. However, this row could not be resolved since c_1, c_3 do not induce $[1, 1]$ and there is another row with three 2's containing 2's in c_1 and c_3. The corresponding hypergraph H_A has one 3-edge, one unforced 2-edge and two forced 2-edges. There is only one graph covers H_A such that it contains only paths of length at most 3, the graph G. In particular, G can be constructed as follows: we have to select forced 2-edges, in 3-edge $\{c_1, c_2, c_3\}$ we select edge (c_1, c_2) (respectively, (c_2, c_3), it will not affect the hypergraph covering), and finally, we do not add unforced 2-edge $\{c_1, c_3\}$ to G. Note that this is a canonical covering. A haplotype matrix B corresponds to this selection of edges in G, hence its conflict graph is G. However, removal of any row from matrix B is not able to eliminate all conflicts in B. Hence, the condition (2) of Theorem 3 is not satisfied and B cannot be explained by a GTN.

Since we have examined all possible ways how to infer a haplotype matrix from A, it follows by Corollary 8 that A cannot be explained by a GTN.

The crucial reason why in the above example A cannot be explained by a GTN even though we can easily find a valid hypergraph covering of the hypergraph of A is the fact that columns c_1, c_2, c_3 induce $[0, 1, 0]$ in A. The following claim captures this property.

Claim 19. *Given a genotype matrix A with the WD property, let B be a haplotype matrix inferred from A which can be explained by a GTN. Let c_1, c_2, c_3 be three SNPs (not necessarily ordered in this way). If they induce $[0, 1, 0]$ in A then the conflict graph G_B cannot contain both edges (c_1, c_2) and (c_2, c_3).*

Proof. Assume to the contrary that both pairs c_1, c_2 and c_2, c_3 conflict in B. Hence, $B[c_1, c_2]$ (respectively, $B[c_2, c_3]$) contains all three pairs $[0, 1]$, $[1, 0]$ and $[1, 1]$. By Observation 9, the pair c_1, c_3 is weakly active in A, hence $B[c_1, c_3]$ contains $[0, 1]$ and $[1, 0]$. Since B can be explained by a GTN, by Theorem 3, SNPs c_1 and c_3 cannot conflict, i.e., $B[c_1, c_3]$ cannot contain pair $[1, 1]$. Hence, $B[c_1, c_2, c_3]$ contains $[1, 0, 0]$, $[1, 1, 0]$, $[0, 0, 1]$ and $[0, 1, 1]$. Since the triple c_1, c_2, c_3 induces $[0, 1, 0]$, $B[c_1, c_2, c_3]$ contains also triple $[0, 1, 0]$. B does not satisfy condition (2) of Theorem 3, a contradiction. □

The following definition is a generalization of Definition 20 which incorporates constrains shown in Claim 19.

Definition 26 (Extended genotype hypergraph). Given an $n \times m$ genotype matrix A with the WD property, the *extended genotype hypergraph* $\bar{H}_A$ of A has the set of SNPs $\{1, \ldots, m\}$ and contains all hyperedges of genotype hypergraph H_A. In addition, it contains ordered 3-edges, called *switches*. For every triple of SNPs c_1, c_2, and c_3 such that there are distinct (forced or unforced) hyperedges e and e' such that $c_1, c_2 \in e$ and $c_2, c_3 \in e'$ and the triple induces $[0, 1, 0]$ in A, H_A contains a switch $[c_1, c_2, c_3]$.

We say that a graph G is a *canonical covering* of extended hypergraph $\bar{H}_A$ if G is a canonical covering for underlying H_A and in addition for every switch (c_1, c_2, c_3), it contains at most one of the edges (c_1, c_2) and (c_2, c_3).

Using Claim 19 we can immediately extend the result in Corollary 8.

Corollary 9. *Given a genotype matrix A with the WD property, if A can be explained by a GTN then there is a haplotype matrix B inferred from A which can be explained by a GTN and its conflict graph is a canonical covering of the extended hypergraph $\bar{H}_A$.*

The hypergraph covering problem for extended genotype hypergraphs can be formulated as follows.

Problem 5 (Extended Hypergraph Covering (EHC) problem). Given an extended hypergraph $\bar{H}$, determine whether it is possible to find a canonical covering graph G for $\bar{H}$ such that each component of G is a path of length at most 3 satisfying the ordered component property. In case it is possible, find such a covering G with the minimum number of edges.

5.1.6 Reduction to the EHC Problem

The following theorem shows that the GTNH problem for genotype matrices with the WD property can be reduced to the EHC problem.

Theorem 23. *Consider a genotype matrix A with the WD property. Then A can be explained by a GTN if and only if there exists a canonical covering G of $\bar{H}_A$ such that each component of G is a path of length at most 3 and satisfies the ordered component property.*

Proof. The forward implication follows easily by Theorem 3, Lemma 21 and Corollary 9. For the converse, consider a canonical covering G of $\bar{H}_A$ such that (i) each component of G is a path of length at most 3 and satisfies the ordered component property, and (ii) it has the minimum number of edges (our off all canonical coverings satisfying condition (i)). Consider a haplotype matrix B corresponding to covering G. Recall that G must be the conflict graph of B.

Since all components are paths of length at most 3 with the ordered component property, condition (1) of Theorem 3 holds. We will show that condition (2) of Theorem 3 is satisfied for each component of G.

Consider a component K of G. If it is a singleton or an edge, condition (2) is trivially satisfied. Second, assume that K is a path of length 2, say (c_1, c_2, c_3). Consider any B defined by any covering G from $\bar{H}_A$. As in the proof of Claim 19, the submatrix $B[K]$ must contain triples $[1,0,0]$, $[1,1,0]$, $[0,1,1]$ and $[0,0,1]$. It can also contain triples $[0,0,0]$ and $[0,1,0]$, all other triples would introduce a conflict between c_1 and c_3. It is easy to see that the component K satisfies condition (2) if and only if $B[K]$ does not contain $[0,1,0]$. We will show that $B[K]$ does not contain $[0,1,0]$.

Assume to the contrary that $B[K]$ contains $[0,1,0]$. It must be inferred from some sequence in $A[K]$. However, since a canonical covering G contains both edges (c_1, c_2) and (c_2, c_3), $\bar{H}_A$ does not contain switch (c_1, c_2, c_3), and hence either c_1, c_2, c_3 do not induce $[0,1,0]$ in A or there are no two distinct hyperedges e and e' in $\bar{H}_A$ such that $c_1, c_2 \in e$ and $c_2, c_3 \in e'$. The second case is not possible as edges (c_1, c_2) and (c_2, c_3) in G require existence of such e and e' in H_A. Assume that c_1, c_2, c_3 do not induce $[0,1,0]$ in A. There are only four possible rows in A which could be resolved to $[0,1,0]$ in B: $[2,2,2]$, $[2,1,2]$, $[2,2,0]$ and $[0,2,2]$. In the first two cases, to infer $[0,1,0]$ in B, we would have to also infer $[1,0,1]$ (in the first case) or $[1,1,1]$ (in the second case). Since the pair c_1, c_3 is weakly active, it would conflict in B, a contradiction. Obviously, the third and fourth cases are symmetric, hence we will consider only one of them. Let r be the row containing $[2,2,0]$ in $A[K]$. To infer $[0,1,0]$ from $[2,2,0]$, the pair must be resolved unequally, i.e.,

$I_B(x_{rc_1c_2}) = 1$. If r contains another 2, say in column c, then by Claim 15, one of the pairs c, c_1 and c, c_2 conflicts in B, which is a contradiction with the fact that K is a connected component of $G_B = G$. Hence, assume that r contains only two 2's. By Definition 25, $I_B(x_{rc_1c_2}) = 0$, i.e., the sequence $[0, 1, 0]$ cannot be inferred in $B[K]$, a contradiction.

B	c_1	c_2	c_3	c_4
	0	0	0	0
$*$	0	0	0	1
	0	0	1	0
$**$	0	0	1	1
	0	1	0	0
$**$	0	1	1	0
$*$	1	0	0	0
?	1	0	0	1
$**$	1	1	0	0

Figure 5.3: All possible sequences (except for the one with the question mark) $B[K]$ can contain where a component K of G_B is a path (c_1, c_2, c_3, c_4). The rows with star(s) are necessarily contained in $B[K]$.

Finally, assume that K is a path of length 3, say (c_1, c_2, c_3, c_4). Consider any B defined by any covering G from $\bar{H}_A$. By Observation 9, the pairs c_1, c_2, c_2, c_3 and c_3, c_4 are active in A, hence the pairs c_1, c_3 and c_2, c_4 are weakly active. Since, the pairs c_1, c_3 and c_2, c_4 do not conflict in B, $B[K]$ can contain only following 9 quadruples: $[0, 0, 0, 0]$, $[0, 0, 0, 1]$, $[0, 0, 1, 0]$, $[0, 0, 1, 1]$, $[0, 1, 0, 0]$, $[0, 1, 1, 0]$, $[1, 0, 0, 0]$, $[1, 0, 0, 1]$ and $[1, 1, 0, 0]$, cf. Figure 5.3. Since, the pairs c_1, c_2, c_2, c_3 and c_3, c_4 conflict in B, the rows with two stars "$**$" are necessarily in $B[K]$. Consequently, $B[c_1, c_4]$ contains pairs $[0, 1]$ and $[1, 0]$ and since c_1, c_4 do not conflict, the row with the question mark "?" cannot appear in $B[K]$. Without a row with "?" in $B[K]$, the rows with one star "$*$" become necessary to guarantee the conflicts between c_1, c_2 and between c_3, c_4. Hence, all rows with star(s) are necessarily in $B[K]$ and rows with no symbol are possibly in $B[K]$. The only candidate for x is 0110. It is easy to see that the condition (2) of Theorem 3 is satisfied if and only if $B[K]$ does not contain any of $[0, 0, 1, 0]$ and $[0, 1, 0, 0]$.

W.l.o.g. assume that $B[K]$ contains $y = [0, 0, 1, 0]$. By Definition 26, the extended hypergraph $\bar{H}_A$ does not contain the switch (c_2, c_3, c_4), and hence c_2, c_3, c_4 do not induce $[0, 1, 0]$ in A. As in the previous case, there are four possible sequences in $A[c_2, c_3, c_4]$ which could be inferred to produce $[0, 1, 0]$ in $B[c_2, c_3, c_4]$, and building on that 8 possible sequences in $A[K]$ which could be inferred to produce y: (a) $[0, 2, 2, 2]$, (b) $[2, 2, 2, 2]$, (c) $[0, 2, 1, 2]$, (d) $[2, 2, 1, 2]$, (e) $[0, 2, 2, 0]$, (f) $[2, 2, 2, 0]$, (g) $[0, 0, 2, 2]$ and (h) $[2, 0, 2, 2]$. Let us analyze all these possibilities:

(a) The only way how to infer y from $[0,2,2,2]$ is to resolve c_2, c_4 equally. This would produce also a quadruple $[0,1,0,1]$ in B, and hence a conflict between c_2 and c_4. Which contradicts the fact that K is a path.

(b) To infer sequences from $[2,2,2,2]$ avoiding any conflict between c_1, c_3, c_2, c_4 or c_1, c_4, the pairs c_1, c_2 and c_3, c_4 must be resolved equally, and other pairs unequally. Hence, the sequence y is not produced.

(c) To infer sequences from $[0,2,1,2]$ avoiding a conflict between c_2, c_4, the pair is resolved unequally, i.e., y is not produced.

(d) The sequences $[2,2,1,2]$ cannot appear in $A[K]$ as otherwise the pair $[1,1]$ is induced by c_1, c_3 in A, i.e., c_1, c_3 would conflict.

(e) Let r be the row containing $[0,2,2,0]$ in A. To produce y, the pair c_2, c_3 has to be resolved unequally. If r contains another 2, say in column c, the one of the pairs c, c_2 and c, c_3 would have to be resolved equally, hence producing a conflict. This would contradict the fact that K is a component in G_B. On the other hand, if r contains only two 2's, by Definition 25, the pair c_2, c_3 is resolved equally in row r, thus not producing y in B.

(f) Let r be a row containing $[2,2,2,0]$ in $A[K]$. First, note that r does not contain any other 2, say in column c. For otherwise there would be a 4-edge in $\bar{H}_A$ containing c_1, c_2, c_3 and c, which would introduce an edge between c and one of c_1, c_2, c_3 in G, a contradiction. There are two ways how to infer sequences from r avoiding conflicts not in K: (i) resolving c_1, c_2 equally; (ii) resolving c_2, c_3 equally. In the case (ii), y is not produced from $[2,2,2,0]$. In the case (i), we will consider another covering from $\bar{H}_A$, which differs from the current one by choosing edge (c_2, c_3) instead of (c_1, c_2) when processing a row containing $[2,2,2,0]$ in $A[K]$. This new covering might be not canonical. Indeed, it is not canonical if r was the only row containing 2's in c_1, c_2 and at least one other 2 such that $I_B(x_{rc_1c_2}) = 0$ and there is no forced 2-edge (c_1, c_2). In such a case, to make the new covering canonical, we also change the inferring of every row containing only two 2's and those in columns c_1 and c_2 from equal to unequal. Obviously, this new covering will not introduce any new conflict, although it will remove conflict between c_1 and c_2. In such a case, we have found another graph G' that canonically covers $\bar{H}_A$ with smaller number of edges, a contradiction with the assumption that G has the minimum number of edges.

(g) Similar to the case (e).

(h) To infer sequences from $[2, 0, 2, 2]$ avoiding any conflict between c_1, c_3 or c_1, c_4, c_3, c_4 is resolved equally, and other pairs unequally. Hence, the sequence y is not produced.

Thus each component of G satisfies condition (2) of Theorem 3 which concludes the proof. □

5.2 The Extended Hypergraph Covering Problem is Intractable

Unfortunately, the EHC problem is intractable in general as proved in the following theorem, and thus Theorem 23 cannot be used to polynomially solve an arbitrary weak diagonal instance of the GTNH problem.

Theorem 24. *The EHC problem is NP-complete.*

Proof. It is easy to see that the EHC problem is in NP. The proof is done by conversion from a special instance of 3-SAT problem. This problem is known to be NP-complete even when restricted to formulas where each clause contains 2 or 3 literals and every variable occurs in exactly 3 clauses — once positive and twice negated [84]. Let $f(x_1, x_2, \ldots, x_m) = C_1 \wedge \cdots \wedge C_k$ be such a formula in conjunctive normal form, where $C_1, \ldots, C_k$ are clauses. Let $p_1, \ldots, p_m$ be all occurrences of literals $\{x_1, \ldots, x_m, \neg x_1, \ldots, \neg x_m\}$ in f. For every $i = 1, \ldots, k$, we have

$$C_i = p_{s_{i,1}} \vee p_{s_{i,2}} \qquad \text{or} \qquad C_i = p_{s_{i,1}} \vee p_{s_{i,2}} \vee p_{s_{i,3}}$$

depending on whether C_i contains 2 or 3 literals. Let $S_i = \{s_{i,1}, s_{i_2}\}$ or $S_i = \{s_{i,1}, s_{i,2}, s_{i,3}\}$, respectively. Sets $S_1, \ldots, S_k$ forms a decomposition of set $\{1, \ldots, m\}$. For every $i = 1, \ldots, m$, let $p_{t_{i,1}}$ be the occurrence of positive literal x_i and $p_{t_{i,2}}, p_{t_{i,3}}$ two occurrences of negated literal $\neg x_i$. Let $T_i = \{t_{i,1}, t_{i,2}, t_{i,3}\}$, Again, sets $T_1, \ldots, T_m$ form a decomposition of set $\{1, \ldots, m\}$. We will construct an extended hypergraph $\bar{H}(f)$ such that it has a canonical covering with all components being paths of length at most 3 and satisfying the ordered-component property if and only if f is satisfiable. The hypergraph will only contain forced 2-edges and 3-edges. Hence, each covering is canonical, and we will not mention this property of the covering in the remaining of the proof. We will form the hypergraph $\bar{H}(f)$ as a union of several hypergraphs, called gadgets, one for each clause, and one for each variable. The numbering of vertices in the hypergraph is important as it influence the ordered-component property. All constructed gadgets will have to types of vertices (SNPs): depicted by a dot and depicted by a cross. For our construction it will be sufficient to number vertices so that all vertices with a cross have higher number than vertices with a dot, which can be easily achieved.

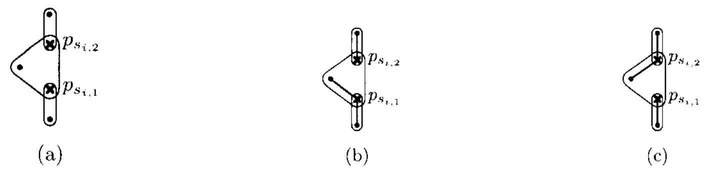


Figure 5.4: Part of hypergraph $\bar{H}(f)$ for clause $C_i = p_{s_{i,1}} \vee p_{s_{i,2}}$ and all possible graphs cover it, each representing one case how the clause can become satisfied. The vertices with cross have higher numbers than vertices with dots.

For every clause with two literals $C_i = p_{s_{i,1}} \vee p_{s_{i,2}}$, we construct a part of hypergraph (a gadget) consisting of two forced 2-edges and one 3-edge as depicted in Figure 5.4. The figure also shows all possible graphs cover the gadget satisfying the conditions of the EHC problem. In these figures, a variable p_j has value 1 if no other edge (from the other parts of $\bar{H}(f)$) can be joining the vertex p_j. For instance, in the first graph, $p_{s_{i,1}}$ has value 1, as any other edge joining $p_{s_{i,1}}$ would increase the degree of this vertex to 3, and $p_{s_{i,2}}$ has value 0. Note that in each hypergraph covering of the gadget at least one of $p_{s_{i,1}}$ and $p_{s_{i,2}}$ has value 1.

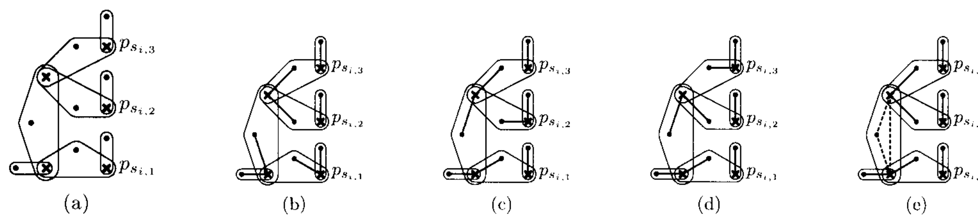


Figure 5.5: (a) The part of hypergraph $\bar{H}(f)$ for clause $C_i = p_{s_{i,1}} \vee p_{s_{i,2}} \vee p_{s_{i,3}}$. (b–d) Three possible graphs that cover from it, each representing one case how the clause can become satisfied. (e) A trial to search for a hypergraph covering with values of $p_{s_{i,1}}, p_{s_{i,2}}, p_{s_{i,3}}$ set to 0.

For every clause with three literals $C_i = p_{s_{i,1}} \vee p_{s_{i,2}} \vee p_{s_{i,3}}$, we construct a gadget consisting of four forced 2-edged and four 3-edges as depicted in Figure 5.5(a). Figure 5.5 also shows three possible graphs (b d) that cover the part of hypergraph satisfying the conditions of the EHC problem. There are other graphs which can be covering for the part of $\bar{H}(f)$, however in each of them at least one of $p_{s_{i,1}}, p_{s_{i,2}}, p_{s_{i,3}}$ has value 1. Indeed, assume that all three values are set to 0. Then no edge from inside of gadget can be joining any of $p_{s_{i,1}}, p_{s_{i,2}}, p_{s_{i,3}}$. We have the situation depicted in Figure 5.5(e). Now, there is no edge to be selected from the 3-edge which vertices are connected with dashed edges without increasing degree of some vertex to 3.

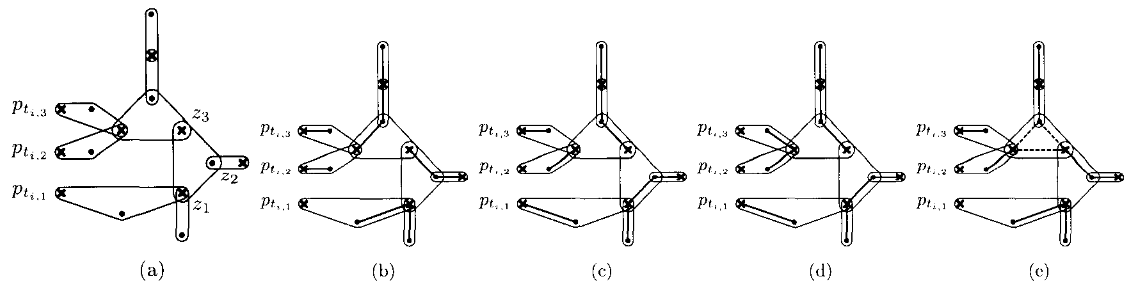


Figure 5.6: (a) The part of hypergraph $\bar{H}(f)$ verifying the values of three occurrences of a variable x_i. (b-d) Three possible hypergraph coverings. In (b), $p_{t_{i,1}}$ has value 1 and forces values of $p_{t_{i,2}}$ and $p_{t_{i,3}}$ to 0. In (c), $p_{t_{i,2}}$ has value 1 and in (d), both $p_{t_{i,2}}$ and $p_{t_{i,3}}$ have value 1. In both cases (c) and (d), $p_{t_{i,1}}$ is forced to have value 0. (e) A trial to search for a hypergraph covering with values of $p_{t_{i,1}}$ and $p_{t_{i,2}}$ set to 1.

The second part of the construction checks whether three occurrences of a variable x_i: $p_{t_{i,1}}, p_{t_{i,2}}, p_{t_{i,3}}$ do not have contradictory values. That is if $p_{t_{i,1}}$ (positive occurrence) has value 1 then both $p_{t_{i,2}}$ and $p_{t_{i,3}}$ (negated occurrences) should have values 0, and if at least one of $p_{t_{i,2}}$ and $p_{t_{i,3}}$ has value 1 then $p_{t_{i,1}}$ should have value 0. This is achieved by a gadget consisting of four forced 2-edges and four 3-edges depicted in Figure 5.6(a). Figures 5.6(b d) show three possible graphs that cover the gadget. In these figures, a variable p_j has value 1 if no edge in the gadget joins p_j, which is in agreement with interpretation of values of p_i's in gadgets of the first part of construction.

Let us verify the claimed property of the gadget. Assume for instance that both $p_{t_{i,1}}$ and $p_{t_{i,2}}$ have values 1. Hence, no edge from inside of the gadget is joining these two vertices. Then to avoid a vertex of degree higher than 2, from the 3-edge $\{z_1, z_2, z_3\}$ we have to select edge (z_2, z_3), cf. Figure 5.6(e). Now, there is no edge to be selected from the 3-edge which vertices are connected with dashed edges without producing a path of length 4 or 5. The other cases can be proved using similar arguments.

Now, let us verify that it is possible to find a hypergraph covering of $\bar{H}(f)$ which satisfies conditions of the EHC problem if and only if f is satisfiable. First, consider a graph G that covers from $\bar{H}(f)$ such that each component is a path of length at most 3. For every clause C_i, at least one of $p_{s_{i,1}}, p_{s_{i,2}}$ (respectively, $p_{s_{i,1}}, p_{s_{i,2}}, p_{s_{i,3}}$) has value 1 in G. Let it be p_{q_i} (if there are several literals in C_i with value 1 in G, pick any of them). We will form a true assignment as follows. For every x_j, if there is $p_{q_i} = x_j$, set $x_j = 1$; if there is $p_{q_i} = \neg x_j$, set $x_j = 0$; otherwise set x_j to any value. As long as we guarantee that there are no i, i' such that $p_{q_i} = x_j$ and $p_{q_{i'}} = \neg x_j$, the above definition is correct and obviously is a true assignment for f. Assume for contrary that $p_{q_i} = x_j$ and $p_{q_{i'}} = \neg x_j$. Obviously, $q_i = t_{j,1}$ and $q_{i'}$ is either $t_{j,2}$ or $t_{j,3}$. Now, since we $p_{t_{j,1}}$ has value 1 and one of

$p_{t_{j,2}}, p_{t_{j,3}}$ has value 1, we have a contradiction with the property of the gadget for x_j.

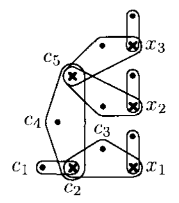
For the converse, consider a true assignment for f. For every clause $C_i = p_{s_{i,1}} \vee p_{s_{i,2}}$ with two literals, there is at least one literal p_{q_i} with value 1, where $q_i \in \{s_{i,1}, s_{i,2}\}$. If $q_i = s_{i,1}$, search for a hypergraph covering of the gadget for C_i as depicted in Figure 5.4(b). If $q_i = s_{i,2}$, as in Figure 5.4(c). Similarly, for every clause $C_i = p_{s_{i,1}} \vee p_{s_{i,2}} \vee p_{s_{i,3}}$ with three literals, there is at least one literal p_{q_i} with value 1, where $q_i \in \{p_{s_{i,1}}, p_{s_{i,2}}, p_{s_{i,3}}\}$. If $q_i = s_{i,1}$ (respectively, $q_i = s_{i,2}$, $q_i = s_{i,3}$), search for a hypergraph covering of the gadget for C_i as depicted in Figure 5.5(b) (respectively, Figure 5.5(c), Figure 5.5(d)). For the gadgets in the second part of construction we will search for coverings as follows. For every x_i, if value of x_i is 1, find a hypergraph covering of the gadget for x_i as depicted in Figure 5.6(b), and if value of x_i is 0, as in Figure 5.6(d). Let G be the union of graphs that cover all gadgets. We will show that G satisfies the conditions of the EHC problem.

First, it is easy to see that all possible edges of G are connecting a vertex with a cross with a vertex with a dot. The ordered-component property follows. It is also easy to see that the components of graphs that cover a single gadget are all paths of length at most 3. Hence, it is enough to verify that p_k-vertices which are shared between gadgets do not combine components to a component which is not a path of length at most 3. Observe that each p_k is shared by exactly two gadgets, one gadget for a clause C_i and one gadget for a variable x_j. In the gadget for C_i, the component containing p_k is either an edge, if p_k is not necessary to satisfy the clause, or a path of length with p_k as the middle vertex, if p_k is necessary to satisfy the clause. In the second case, by definition of G, p_k has value 1 in the considered true assignment for f. In the gadget for x_j, the component containing p_k is either a singleton, if p_k has value 1, and an edge, if p_k has value 0. The only way how to obtain an invalid component is to combine the middle vertex of path of length 2 with an edge, but this will never happen as the first one requires the value of p_k to be 1 and the other to be 0. □

Even though we have proved that the EHC problem is NP-complete, it does not imply the NP-completeness of the GTNH problem. Indeed, it is not possible to construct a genotype matrix resulting in the constructed gadgets for a given boolean formula. The following example shows why.

Example 6. Suppose that there is a boolean formula $f = x_1 \vee x_2 \vee x_3$. We show that it is impossible to construct a genotype matrix A such that the hypergraph H_A for A is isomorphic to the constructed gadgets used in the proof of Theorem 24. In Figure 5.7, We show the gadget for the 3-clause $x_1 \vee x_2 \vee x_3$ and the part of values that A must contain.

Indeed, the three 2's in r_1 and r_2 are added for the hyperedges $\{c_2, c_3, x_1\}$ and



A	c_1	c_2	c_3	c_4	c_5	x_1
r_1	0	2	2	0	0	2
r_2	0	2	0	2	2	0

Figure 5.7: Gadget for clause $x_1 \vee x_2 \vee x_3$ and the part of values that A must contain.

$\{c_2, c_4, c_5\}$ in the gadget respectively. Because of the fact that column pairs $[c_2, c_3]$ and $[c_2, c_4]$ are active, column pair $[c_3, c_4]$ is weakly active. Therefore, to avoid creating conflict between c_3 and c_4, they cannot induce [1, 1] from A. Thus, in r_1 of A, c_4 must have value 0. Similarly, columns c_1, c_5 must be 0 in r_1 and columns c_1, c_3 and x_1 must be 0 in r_2. However, this makes the three columns c_1, c_2 and c_3 induce [0, 1, 0]. According to Definition 26, H_A have to contain a switch $[c_1, c_2, c_3]$. Since $\{c_1, c_2\}$ is a forced 2-edge, (c_2, c_3) cannot be added to any possible covering of H_A. Similarly, (c_2, x_1) cannot be in any covering for H_A either. This makes the edge (c_3, x_1) must be added to any covering of H_A, a contradiction.

However, our recent research shows that the weak diagonal instance of the GTNH problem is the key for proving NP-completeness of the GTNH problem. We present the result in the next section.

5.3 GTNH Problem is NP-Complete

5.3.1 Definitions

We define a special type of genotype matrices below.

Definition 27 (Simple genotype matrix). We say that a genotype matrix is *simple* if every row contains either zero or three 2's.

For clarity, we will consider the following (stripped) version of the extended genotype hypergraph.

Definition 28 (Extended genotype hypergraph (EGH)). An *extended genotype hypergraph* is a hypergraph with m vertices and hyperedges containing either two or three vertices. In addition, an extended genotype hypergraph contains a list of switches, which are ordered triples of vertices.

Given a simple $n \times m$ genotype matrix A, the *extended genotype hypergraph* H_A of A has the set of characters (columns) $\{1, \ldots, m\}$ as a vertex set. Hypergraph H_A contains only the following hyperedges:

- for every row r of A containing three 2's, say in columns c_1, c_2, c_3 there is a hyperedge $e_r = \{c_1, c_2, c_3\}$; and
- for every two columns c_1 and c_2 inducing $[0,1]$, $[1,0]$ and $[1,1]$ in A, there is a hyperedge $\{c_1, c_2\}$ in H_A.

Furthermore, for every triple of characters c_1, c_2, c_3 such that there are distinct hyperedges e and e' such that $c_1, c_2 \in e$ and $c_2, c_3 \in e'$ and the triple induces $[0,1,0]$ in A, H_A contains a switch $[c_1, c_2, c_3]$.

The following definition defines a graph covering of an extended genotype hypergraph.

Definition 29 (Covering of EGH). Consider an extended genotype hypergraph H. We say that a graph G with the same vertex set as H covers H 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) in 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; and

and for every switch $[c_1, c_2, c_3]$, at most one of the edges (c_1, c_2) and (c_2, c_3) is in G. A graph G that covers H is called a *covering* of H.

The EGHC problem is a simpler version of the EHC problem we defined in subsection5.1.5 and it can be formulated as follows:

Problem 6 (Extended Genotype Hypergraph Covering Problem). Given an extended genotype hypergraph H, determine whether there is a covering G of H such that each connected component K of G is a path of length at most 3 satisfying the ordered component property.

The following characterization was shown in the previous section.

Theorem 25. *Consider a simple genotype matrix with the weak diagonal property. Then A can be explained by a galled-tree network if and only if there exists a covering G of H_A such that every component of G is a path of length at most 3 and has the ordered-component property.*

In Section 5.2, it was also shown that the EGHC problem is NP-complete, hence this characterization fails to provide a polynomial solution for the GTNH problem even for such special genotype matrices. On the other hand, since not every extended genotype hypergraph has a corresponding genotype matrix, in particular, the gadgets used to show

NP-completeness of the EGHC problem in Section 5.2 do not, the NP-completeness of EGHC problem does not imply that the GTNH problem is NP-complete. In the next section, we consider special instances of extended genotype hypergraphs for which, as we will see later, it is possible to construct a corresponding genotype matrix and show that the EGHC problem for them remains NP-complete.

5.3.2 GTNH Problem is NP-complete

The proof of NP-completeness is done in two steps. First, we define special instances of extended genotype hypergraphs, called ***natural instances*** and show that the EGHC problem is NP-complete for them. Then we show that it is possible to construct a genotype matrix for each such instance. The NP-completeness of the GTNH problem then follows by the characterization obtained in Section 5.2, (Theorem 25).

5.3.3 Natural Extended Genotype Hypergraph Covering Problem is NP-Complete

Definition 30 (Natural EGH). We say that an extended genotype hypergraph H is *natural* if for any two hyperedges e, e' of H, $|e \cap e'| \leq 1$ and the list of switches contains all and only the following switches: for every vertex c of H with degree at least 3, and for every two hyperedges e_1 and e_2 containing c, and for every two vertices $c_1 \in e_1$ and $c_2 \in e_2$ such that c, c_1, c_2 are all distinct, there is a switch $[c_1, c, c_2]$.

We will show that the natural extended genotype hypergraph covering problem is NP-complete by reduction from 3-SAT. The proof is somewhat similar to the proof of NP-completeness of the EGHC problem [42] illustratedin in Section5.2.

Theorem 26. *The natural extended genotype hypergraph covering problem is NP-complete.*

Proof. The proof is done by a reduction from a special instance of the 3-SAT problem in which each clause contains two or three literals and every variable occurs in exactly three clauses — once positive and twice negated [84]. Let $f(x_1, x_2, \ldots, x_m) = C_1 \wedge \cdots \wedge C_k$ be such a formula in the conjunctive normal form, where $C_1, \ldots, C_k$ are the clauses. Let $p_1, \ldots, p_{3m}$ be the list of all occurrences of literals in f such that $p_{3i-2} = x_i$ and $p_{3i-1} = p_{3i} = \neg x_i$. Now every clause C_i can be written as

$$C_i = p_{s_{i,1}} \vee p_{s_{i,2}} \qquad \text{or} \qquad C_i = p_{s_{i,1}} \vee p_{s_{i,2}} \vee p_{s_{i,3}}$$

depending on whether C_i contains two or three literals.

Next, we construct a natural extended genotype hypergraph $H(f)$ for f which has a covering if and only if the formula f is satisfiable. The hypergraph $H(f)$ will be an edge-disjoint union of several gadgets, one for each clause and one for each variable. The only vertices in common among gadgets will be the literal vertices, in particular, each literal vertex will be shared between one clause gadget and one variable gadget. Furthermore, in each gadget we will mark every vertex either with a dot or a cross such that every literal vertex will be marked with a dot. This will guarantee that our marking will be consistent in whole $H(f)$. Using this marking we order the vertices of $H(f)$ such that every vertex marked with a dot precedes every vertex marked with a cross. This ordering implies that to verify the ordered component property of a covering it is enough to check that in such a covering every path of length two or three alternates between vertices with crosses and dots.

Now we define the gadgets; we start with the clause gadgets. Consider a covering c of $H(f)$. We say that a literal p_j has value 1 in this covering, if c restricted to the clause gadget containing p_j contains an edge incident to the vertex p_j. Note that this is well defined since for every literal vertex p_j there is a unique clause gadget containing it.

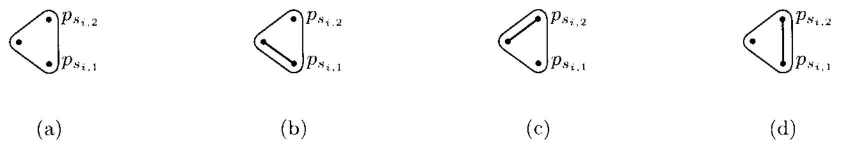


Figure 5.8: The clause gadget for clause $C_i = p_{s_{i,1}} \vee p_{s_{i,2}}$ with two literals and all possible coverings. The coverings are depicted as solid edges joining particular pair of vertices. In (b), $p_{s_{i,1}}$ has value 1 and $p_{s_{i,2}}$ has value 0, in (c) $p_{s_{i,1}}$ has value 0 and $p_{s_{i,2}}$ has value 1, and in (d) both have value 1.

For every clause $C_i = p_{s_{i,1}} \vee p_{s_{i,2}}$ with two literals, we construct a gadget consisting of one 3-edge as depicted in Figure 5.8(a). Figure 5.8(b–d) shows all possible coverings of the gadget. Note that in each such covering, at least one of literals $p_{s_{i,1}}$ and $p_{s_{i,2}}$ has value 1.

For every clause $C_i = p_{s_{i,1}} \vee p_{s_{i,2}} \vee p_{s_{i,3}}$ with three literals, we construct a gadget consisting of one 2-edge and nine 3-edges as depicted in Figure 5.9(a). Figure 5.9(b–d) shows three possible coverings of the gadget, in which exactly one of the literals is set to 1. Note that in our proof, the restriction of a covering corresponding to a satisfiable assignment of the formula f to the gadget will be one of these three coverings. In any other covering of the gadget, an important property is that at least one of the literals has value 1. Indeed, assume that all three values are set to 0. Then in every covering of the gadget in which the condition for switches is satisfied, there is a path of length 4, cf.

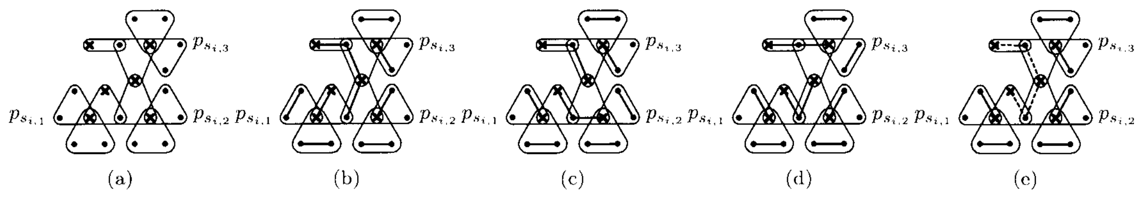


Figure 5.9: (a) The part of hypergraph $\bar{H}(f)$ for clause $C_i = p_{s_{i,1}} \vee p_{s_{i,2}} \vee p_{s_{i,3}}$. (b d) Three possible graph coverings, each representing one case how the clause can become satisfied. (e) The case when all three literals $p_{s_{i,1}}, p_{s_{i,2}}, p_{s_{i,3}}$ are set to 0 leads to a path of length 4 (dashed).

Figure 5.9(e), hence it is not a covering. This guarantees that in any covering of $H(f)$, in the corresponding assignment, every clause of f will be satisfied.

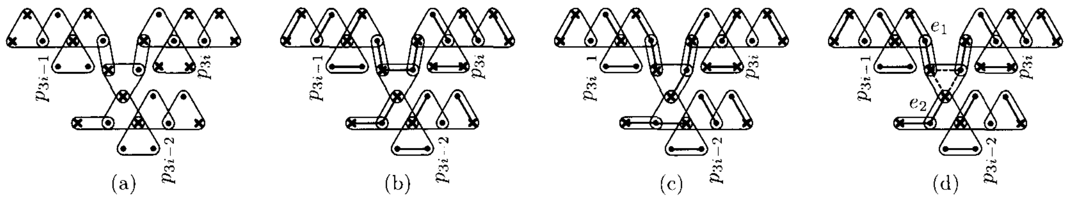


Figure 5.10: (a) The part of hypergraph $H(f)$ verifying the values of three occurrences of a variable x_i. (b-c) Two possible coverings. In (b), we have depicted the unique covering if it is assumed that p_{3i-2} has value 1 (set by clause gadget). As can be seen, this forces values of p_{3i-1} and p_{3i} to 0 (in their clause gadgets). In (c), p_{3i-2} is forced to have value 0 (in its clause gadgets) and p_{3i-1} and p_{3i} can have arbitrary values. In (d), the case when p_{3i-2} and p_{3i-1} are set to 1 leads to a path of length 4 or 5 (depending on which of the dashed edges is chosen).

In the second part of the construction, for each variable x_i, we add a variable gadget which will guarantee that three occurrences of a variable x_i: $p_{3i-2}, p_{3i-1}, p_{3i}$ must be assigned consistent values. That is if p_{3i-2} (positive occurrence) has value 1 then both p_{3i-1} and p_{3i} (negated occurrences) should have values 0, and if at least one of p_{3i-1} and p_{3i} has value 1 then p_{3i-2} should have value 0. This is achieved by a gadget consisting of three 2-edges and thirteen 3-edges depicted in Figure 5.10(a). Figures 5.10(b c) show three possible coverings of the gadget. In these figures, a variable p_j has value 1 if no edge in the gadget joins p_j, which is in agreement with interpretation of values of p_i's in gadgets of the first part of construction.

Let us verify the claimed property of the gadget. Assume for instance that both p_{3i-2} and p_{3i-1} have value 1. No edge covering the variable gadget can be adjacent to any of these two vertices, otherwise the condition on switches (crossing from variable to clause gadgets) would be violated. Hence, the edges connecting the other two vertices of the 3-edges containing p_{3i-2} or p_{3i-1} have to be in the covering. Similarly, edges e_1, e_2 in

Figure 5.10(e) have to be in the covering. Now, there is no edge to be selected to cover the 3-edge in the middle of the gadget, as the selection of any of the dashed edges would produce a path of length 4 or 5. The other cases can be proved using similar arguments.

Finally, we have to check that it is possible to find a covering of $H(f)$ which satisfies the conditions of the EGHC problem (a solution to the EGHC problem) if and only if f is satisfiable. First, consider a graph G that is the solution to the EGHC problem for $H(f)$. For every clause C_i, at least one of $p_{s_{i,1}}, p_{s_{i,2}}$ (respectively, $p_{s_{i,1}}, p_{s_{i,2}}, p_{s_{i,3}}$) has value 1 in G. Let it be p_{q_i} (if there are several literals in C_i with value 1 in G, pick any of them). We will form a true assignment as follows. For every x_j, if there is $p_{q_i} = x_j$, set $x_j = 1$; if there is $p_{q_i} = \neg x_j$, set $x_j = 0$; otherwise set x_j to any value. As long as we guarantee that there are no i, i' such that $p_{q_i} = x_j$ and $p_{q_{i'}} = \neg x_j$, the above definition is correct and obviously is a true assignment for f. Assume that on the contrary, $p_{q_i} = x_j$ and $p_{q_{i'}} = \neg x_j$. Obviously, $q_i = p_{3j-2}$ and $q_{i'}$ is either p_{3j} or p_{3j-1}. Now, since p_{3j-2} has value 1 and one of p_{3j}, p_{3j-1} has value 1, we have a contradiction with the property of the variable gadget for x_j.

For the converse, consider a true assignment for f. For every clause $C_i = p_{s_{i,1}} \vee p_{s_{i,2}}$ with two literals, there is at least one literal in $q_i \in \{s_{i,1}, s_{i,2}\}$ with value 1 in this assignment. If it is $p_{s_{i,1}}$ (respectively, $p_{s_{i,2}}$), pick the covering of the clause gadget for C_i as depicted in Figure 5.8(b) (respectively, Figure 5.8(c)). Similarly, for every clause $C_i = p_{s_{i,1}} \vee p_{s_{i,2}} \vee p_{s_{i,3}}$ with three literals, pick the covering as depicted in Figures 5.9(b d), depending on which literal has value 1 (if there are several pick one). For the variable gadgets we will select the coverings as follows. For every x_i, if value of x_i is 1, pick a hypergraph covering of the gadget for x_i depicted in Figure 5.10(b), and if value of $\neg x_i$ is 1, in Figure 5.10(d). Let G be the union of graphs that cover all gadgets. Now, we need to show that G satisfies the conditions of the EGHC problem. The coverings of each gadgets satisfies these conditions. Since, the only common vertices among gadgets are literal vertices, which have degree 3, the switches with literal vertices in the middle forbid the connected components of the coverings of different gadgets to connect. Hence, G satisfies the conditions of the EGHC problem as well. □

5.3.4 Construction of Genotype Matrix

In this subsection we show that for every natural EGH there is a corresponding genotype matrix. First, let us define the correspondence between genotype matrices and extended genotype hypergraphs.

Lemma 27. *For every natural EGH H, there is a genotype matrix A with zero or three*

2's in every row such that $H_A = H$.

Proof. Let H be a natural EGH. Construct a genotype matrix $A(H)$ using the following steps:

(1) Let $A(H)$ have $|V(H)|$ columns and each vertex $c \in V(H)$ corresponds to a column c of $A(H)$.

(2) For each 3-edge $\{c_1, c_2, c_3\}$ of H, add a new row r to $A(H)$ such that $r[c] = 2$, if $c \in \{c_1, c_2, c_3\}$, and $r[c] = 0$, otherwise.

(3) For each 2-edge $\{c_1, c_2\}$ of H, add a new row r to $A(H)$ such that $r[c] = 1$, if $c \in \{c_1, c_2\}$, and $r[c] = 0$, otherwise.

(4) For every vertex c of H with degree 1, add a new row r to $A(H)$ such that $r[c] = 1$, and $r[c'] = 0$ for any $c' \neq c$.

Let $A = A(H)$. Now, we have to show that $H_A = H$Obviously, H_A and H have the same sets of 3-edges. Consider a 2-edge $\{c_1, c_2\}$ of H. By definition of A, c_1, c_2 induce $[1, 1]$ in A. Let us show, for instance, that they also induce $[1, 0]$. First, if the degree of c_1 in H is one, then there is a special row in A for c_1 which induces $[1, 0]$. Second, assume that there is other hyperedge in H containing c_1. Then the row in A for this hyperedge induces $[1, 0]$. Hence, H_A contains hyperedge $\{c_1, c_2\}$. Finally, observe that H_A cannot contain any other 2-edge, as pair $[1, 1]$ can be induced only in the rows added in the step (3).

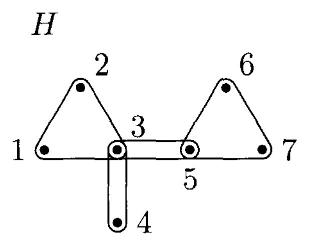
Next, we need to show that H_A and H have the same lists of switches. Assume that H contains a switch $[c_1, c, c_2]$. Then c has a degree at least 3 in H, and there are two distinct hyperedges e_1 and e_2 containing c and $c_1 \in e_1$, $c_2 \in e_2$ such that c, c_1, c_2 are all distinct. Since $d(c) \geq 3$, there is another hyperedge e_3 containing c. Since, $c_1, c_2 \notin e_3$, c_1, c, c_2 induces $[0, 1, 0]$. Hence, by definition, H_A contains the switch $[c_1, c, c_2]$.

Assume now that H_A contains a switch $[c_1, c_2, c_3]$. By definition, there are two distinct hyperedges e and e' such that $c_1, c_2 \in e$ and $c_2, c_3 \in e'$. Since the triple c_1, c_2, c_3 induces $[0, 1, 0]$ and $d(c_2) \geq 2$, there must be another hyperedge e'' containing c_2, but not containing c_1 and c_3. Hence, $d(c_2) \geq 3$, and $[c_1, c_2, c_3]$ is a switch in H. □

Example 7. In this example, we built a genotype matrix A for a natural extended hypergraph H. See Figure 5.11.

5.3.5 Completion of the Proof

To complete the proof we need to show that for every natural EGH H the genotype matrix $A(H)$ constructed in the previous section has the weak diagonal property.



$A(H)$	c_1	c_2	c_3	c_4	c_5	c_6	c_7
	2	2	2	0	0	0	0
	0	0	0	0	2	2	2
	0	0	1	1	0	0	0
	0	0	1	0	1	0	0
	1	0	0	0	0	0	0
	0	1	0	0	0	0	0
	0	0	0	1	0	0	0
	0	0	0	0	0	1	0
	0	0	0	0	0	0	1

Figure 5.11: Example of a genotype matrix $A(H)$ constructed for a natural extended hypergraph H.

Lemma 28. *Let H be a natural extended genotype hypergraph. Then $A(H)$ has the weak diagonal property.*

Proof. In this proof, we show a stronger result that any two columns $c_i, c_j \in A(H)$ induce both [0,1] and [1,0]. First, suppose vertex $c \in H$ has degree more than 1. For any other vertex c', there must exist a hyperedge e such that c is incident on e while c' is not. Indeed, any two hyperedges in H share at most one vertex. From the constructing procedure of $A(H)$ we know that there must be a row r with $r[c] = 2$ (or $r[c] = 1$) and $r[c'] = 0$. In other words, columns c and c' induce [1,0].

Now, for any pair of vertices c and c', if each of them has degree more than 1, then the corresponding pair of columns $c, c' \in A(H)$ induce both [1,0] and [0,1]. If one of them, say c, has degree 1, then according to the last step of the construction of $A(H)$, the two columns induce both [0,1] and [1,0] as well. □

The main result follows by Theorems 25 and 26, and Lemmas 27 and 28.

Theorem 29. *The galled-tree network haplotyping problem is NP-complete.*

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Despite the great progress biomedical research has made, the root causes of common human diseases remain largely unknown. It is widely believed that mutations in DNA sequences, especially those interfering the composition of genes, can cause illness. While research in monogenic diseases have been successful, it becomes more complicated in determining the genetic basis of common "multi-factorial" diseases, which are the result of multiple genetic variants and environmental factors. Haplotypes are believed to be promising genetic markers to tackle the problem. The understanding of haplotype structures will provide fundamental insights into the pathogenesis, diagnosis and treatment of common diseases.

This thesis work is intended to tackle the haplotyping problem using computational methods. In particular, we are interested in inferring haplotypes from genotypes, based on the assumption that the evolutionary history for the original hapltypes satisfies galled-tree network model. After the problem was introduced by Gusfield et al. in 2004 ([54]), not many results have been published related to the problem. In this thesis, we contributed in the following aspects of the problem. We characterized the existence of galled-tree networks for haplotype matrices. As the general galled-tree network haplotyping problem is quite difficult, we studied some special cases of the problem. For one special case of the GTNH problem, where we simplified the galled-tree networks (the simple galled-tree network) that can explain the original haplotypes and we required the input genotype matrices satisfy a combinatorial property (the weak property), we gave a polynomial algorithm to solve it. For another special case of the GTNH problem, where we require the input genotype matrices to satisfy another combinatorial property (the WD property) and no restriction on the galled-tree network model, we characterized it. Finally, based on the previous characterization, we show that the GTNH problem is NP-complete.

6.2 Future Work

For haplotype inferring problem, galled-tree network model can be applied to infer haplotypes from wider range of genotype data than the perfect phylogeny model, as when there is no recombination happens GTN is exactly perfect phylogeny. Furthermore, its feature that allows a small number of recombinations in each haplotype block favors the biological observation of human chromosomes. It is interesting to ask for an efficient algorithm for haplotype inferring using the galled-tree network model.

6.2.1 Design Efficient Algorithms for some Restricted Versions of the GTNH Problem

Now that we know the general galled-tree network haplotyping problem is NP-Complete, simpler versions of the problem are intriguing to explore. Song et al ([94]) studied the problem which only allows one recombination to occur in a galled-tree network and with the assumption that PPH solution, if there is any, for each sub-matrix of the input genotype matrix is unique. One open problem is to find an efficient algorithm for GTNH problem with the output galled-tree network with exactly one gall and there is no limitation on the number of PPH solutions for each sub-matrix M_L and M_R. Furthermore, it is interesting to look at the GTNH problem by restricting the size for each recombination cycle instead of restricting the number of recombinations. In this thesis, we described a polynomial algorithm for the case when the galled-tree networks that can explain the original haplotypes have only size-2 recombination cycles. It will be interesting to see for size-3 galls, if we can still find a polynomial algorithms to solve this special problem.

6.2.2 GTNH Problem with Partially Known Haplotypes or Galled-Tree Networks

Another problem that is interesting to explore is to utilize partially known information when trying to infer haplotypes from genotypes. Such partial information includes part of the haplotypes that form the genotypes in question and part of the galled-tree structure which can be utilized to explain the evolutionary history for the original haplotypes. For instance, the partial haplotype information can come from some molecular haplotyping methods. Though there are no efficient molecular haplotyping methods to sequence long-range haplotypes so far, some sequencing technique is still used to sequence short range haplotype information. If such information can be used in inferring longer range of genotype data, it would save lots of costs and energy and also increase the confidence about the correctness of the results from the computational methods. Partial galled-tree network structure or partial phylogeny tree structure can help in reducing possible number of solutions and furthermore increase the accuracy of the inferring.

6.2.3 Applications

It would be of great interests to the researchers of genetic studies if the polynomial algorithm described in the thesis can be applied to real-life biological data. Such data can be gathered from the HapMap project website, where genotype data of SNPs with high density for individuals from several origins have been sequenced ([57]). Before applying the algorithm to infer haplotypes from genotypes, we need to first verify if the data satisfy weak property or find blocks of SNPs such that SNPs in each block across all individuals satisfying the property. Next, for the data blocks that satisfy weak property, apply our algorithm to infer haplotypes.

Recall that the galled-tree network haplotyping problem together with the perfect phylogeny haplotyping problem both rely on an observation that many human chromosomes are blocky, and nucleotides inside each block tend to inherit together. Without assumption on boundaries of these "blocks", our algorithm is likely to fail when used to infer haplotypes.

There are several ways we can apply our algorithm in inferring large-scale genotype data. Suppose we have a set of genotypes G with length l for each genotype in G. Let t and t_0 be two pointers such that t_0 points to the beginning of the SNP block in consideration and t points to the current SNP position of the genotypes. Set the initial value of t_0 to be 1. Increase t by 1 every time we read one more SNP from each genotype in G. As long as the block of SNPs between position t_0 and t can be solved using our algorithm, increase t again. When t reaches a point, where the genotypes cannot be inferred using our algorithm, record the inferred values, and reset the pointer t_0 and t to start from the current position. Repeat the above procedures, until all the SNP values in G are inferred. This strategy not only inferred the genotypes, but also found some boundaries for possible haplotype blocks, where SNPs in each block are formed by haplotypes whose evolutionary history satisfies galled-tree network model. In order to get haplotypes for the whole genotypes in G, we need to have overlaps between haplotype blocks. For instance, after finding a block boundary using the above procedure, reset t_0 and t to the last position of the previous block and start the searching again. Inferring of the new block should be consistent with the inferring of the first SNP from the previous procedure. Note that

Eskin et al. ([32]) illustrated a way of how to joining together inferrings of haplotypes from adjacent genotype blocks. Their algorithm HAP was utilized successfully in inferring haplotypes for chromosome size samples.

Another way to utilize our algorithm in inferring haplotypes, would be to use existing block finding algorithms to find out possible block boundaries, and then apply our algorithm in each block. This strategy might not work if there are blocks where the evolutionary history for the original haplotypes do not satisfy galled-tree network model. But again, for these cases, we can apply our strategy above to do the inferring for sub-blocks.

Galled-tree network model can be applied in searching for recombination hotspots (genomic regions that have much higher recombination rates than its neighboring regions) as well. There is some research going on in searching for hotspots using general phylogenetic network models with minimum number of recombinations ([115, 5]). Though galled-tree network only allows non-interacting recombinations, our preliminary experimental results show that using the model we can correctly detect experimentally verified hotspots on some biological data sets ([67]). It is interesting to explore whether GTN can be applied in detecting hotspots in other biological data sets.

Bibliography

[1] J. Aerssens, M. Armstrong, R. Gilissen, and N. Cohen. The human genome: an introduction. *The Oncologist*, 6:100–109, 2001.

[2] R. Agarwala and D. Fernández-Baca. A polynomial-time algorithm for the perfect phylogeny problem when the number of character states is fixed. *SIAM Journal on Computing*, 23:1216–1224, 1994.

[3] R. Agarwala and D. Fernandez-Baca. Simple algorithms for perfect phylogeny and riangulating colored graphs. *International Journal of Foundations of Computer Science*, 7:11–21, 1996.

[4] D. Altshuler, J. N. Hirschhorn, M. Klannemark, C. M. Lindgren, M. Vohl, J. Nemesh, C. R. Lane, S. F. Schaffner, S. Bolk, C. Brewer, T. Tuomi, D. Gaudet, T. J. Hudson, M. Daly, L. Groop, and E. S. Lander. The common PPARγ Pro12Ala polymorphism is associated with decreased risk of type 2 diabetes. *Nature Genetics*, 26:76–80, 2000.

[5] V. Bafna and V. Bansal. Improved recombination lower bounds for haplotype data. In *RECOMB '05: Proceedings of the Ninth Annual International Conference on Research in Computational Molecular Biology*, Lecture Notes in Computer Science, pages 569–584. Springer Berlin / Heidelberg, 2005.

[6] V. Bafna, D. Gusfield, S. Hannenhalli, and S. Yooseph. A note on efficient computation of haplotypes via perfect phylogeny. *Journal of computational biology*, 11:858–866, 2004.

[7] V. Bafna, D. Gusfield, S. Hannenhalli, and S. Yooseph. A note on efficient computation of haplotypes via perfect phylogeny. *Journal of Computational Biology*, 11:858–866, 2004.

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

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

[10] H. Bodlaender, M. Fellows, and T. Warnow. Two strikes against perfect phylogeny. In *Proceedings of the 19th International Colloquium on Automata, Languages, and Programming*, Lecture Notes in Computer Science, pages 273–283. Springer Verlag, 1992.

[11] M. S. Bradshaw, J. A. Bollekens, and F. H. Ruddle. A new vector for recombination-based cloning of large DNA fragments from yeast artificial chromosomes. *Nucleic Acids Research*, 23:4850–4856, 1995.

[12] P. O. Brown and L. Hartwell. Genomics and human disease – variations on variation. *Nature genetics*, 18:91–93, 1998.

[13] C. Burgtorf, P. Kepper, M. Hoehe, C. Schmitt, R. Reinhardt, H. Leharch, and S. Sauer. Clone-based systematic haplotyping (CSH): a procedure for physical haplotyping of whole genomes. *Genome Research*, 13:2717–2724, 2003.

[14] A. M. Campbell and L. J. Heyer. *Discovering genetics, proteomics, & bioinformatics.* Pearson Education, Inc., second edition, 2007.

[15] A. Chakravarti. Single Nucleotide Polymorphisms...to a future of genetic medicine. *Nature*, 409:822–823, 2001.

[16] R. Chung and D. Gusfield. Perfect phylogeny haplotyper: haplotype inferral using a tree model. *Bioinformatics*, 19:780–781, 2003.

[17] A. G. Clark. Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular Biology Evolution*, 7:111–122, 1990.

[18] The International HapMap Consortium. The international hapmap project. *Nature*, 426:789–796, 2003.

[19] D. C. Crawford and D. A. Nickerson. Definition and clinical importance of haplotypes. *Annual Review of Medicine*, 56:303–320, 2005.

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

[21] W. H. Day and D. Sankoff. Computational complexity of inferring phylogenies by compatibility. *Systematic Zoology*, 35:224–229, 1986.

[22] S. S. Deeb, L. Fajas, M. Nemoto, J. Pihlajamäki, L. Mykkänen, J. Kuusisto, M. Laakso, W. Fujimoto, and J. Auwerx. A pro12ala substitution in pparγ2 associated with decreased receptor activity, lower body mass index and improved insulin sensitivity. *Nature Genetics*, 20:284–287, 1998.

[23] C. Ding and C. R. Cantor. Direct molecular haplotyping of long-range genomic DNA with M1-PCR. *Proceedings of the Nationlal Academy of Sciences of the United States of America*, 100:7449–7453, 2003.

[24] Z. Ding, V. Filkov, and D. Gusfield. A linear-time algorithm for the perfect phylogeny haplotyping (PPH) problem. In *RECOMB '05: Proceedings of the Ninth Annual International Conference on Research in Computational Molecular Biology*, pages 585–600. Springer Berlin / Heidelberg, 2005.

[25] J. S. Dorman, R. E. LaPorte, R. A. Stone, and M. Trucco. Worldwide differences in the incidence of type I diabetes are associated with amino acid variation at position 57 of the HLA-DQ beta chain. *Proceedings of the National Academy of Sciences*, 87:7370–7374, 1990.

[26] J. A. Douglas, M. Boehnke, E. Gillanders, J. M. Trent, and S. B. Gruber. Experimentally-derived haplotypes substantially increase the efficiency of linkage disequilibrium studies. *Nature Genetics*, 28:361–364, 2001.

[27] F. F. Dragan. Strongly orderable graphs: A common generalization of strongly chordal and chordal bipartite graphs. *Discrete Applied Mathematics*, 99:427–442, 2000.

[28] A. Dress and M. Steel. Convex tree realizations of partitions. *Applied Mathematics Letters*, 5:3–6, 1992.

[29] C. M. Drysdale, D. W. McGraw, C. B. Stack, J. C. Stephens, R. S. Judson, K. Nandabalan, K. Arnold, G. Ruano, and S. B. Liggett. Complex promoter and coding region $\beta 2$ -adrenergic receptor haplotypes alter receptor expression and predict in vivo responsiveness. *Proc. Natl. Acad. Sci. USA*, 97:10483–10488, 2000.

[30] E. Eskin, E. Halperin, and R. M. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. *Journal of Bioinformatics and Computational Biology*, 1:1 20, 2003.

[31] E. Eskin, E. Halperin, and R. M. Karp. Large scale reconstruction of haplotypes from genotype data. In *RECOMB '03 Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology*, pages 104–113, Berlin, Germany, 2003. ACM press New York, NY. USA.

[32] E. Eskin, E. Halperin, and R.M. Karp. Large scale reconstruction of haplotypes from genotype data. In *RECOMB '03: Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology*, pages 104–113, New York, NY, USA, 2003. ACM Press.

[33] L. Excoffier and M. Slatkin. Maximum likelihood estimation of molecular haplotype frequencies in a diploid population. *Molecular Biology Evolution*, 12:921–927, 1995.

[34] D. Fernández-Baca. The Perfect Phylogeny Problem. In *Steiner Trees In Industries*. Kluwer Academic Publishers, 2000.

[35] D. Fernández-Baca and J. Lagergren. A polynomial-time algorithm for near-perfect phylogeny. In *Proceedings of the 23rd International Conference on Automata, Laguages, and Programming*, Lecture Notes in Computer Science, pages 670–680. Springer-Verlag, 1996.

[36] L. Fugger, N. Morling, L. P. Ryder, N. Odum, and A. Svejgaard. Technical aspects of typing for hla-dp alleles using allele-specific dna in vitro amplification and sequence-specific oligonucleotide probes. detection of single base mismatches. *Journal of immunological methods*, 129:175–185, 1990.

[37] S. V. Muse G. Gibson. *A primer of genome science.* Sunderland, MA:Sinauer Associates, 2004.

[38] S. B. Gabriel, S. F. Schaffner, H. Nguyen, J. M. Moore, J. Roy, B. Blumenstiel, J. Higgins, M. DeFelice, A. Lochner, M. Faggart, S. N. Liu-Cordero, C. Rotimi, A. Adeyemo, R. Cooper, R. Ward, E. S. Lander, M. J. Daly, and D. Altshuler. The structure of haplotype blocks in the human genome. *Science*, 296:2225–2229, 2002.

[39] M. R. Garey and D. S. Johnson. *Computers and intractability : a guide to the theory of NP-completeness.* Freeman, 1979.

[40] S. Gretarsdottir, G. Thorleifsson, S. T. Reynisdottir, A. Manolescu, S. Jonsdottir, T. Jonsdottir, T. Gudmundsdottir, S. Bjarnadottir, O. B Einarsson, H. Gudjonsdottir, M. Hawkins, G. Gudmundsson, H. Gudmundsdottir, H. Andrason, A. Gudmundsdottir, M. Sigurdardottir, T. Chou, J. Nahmias, S. Goss, S. Sveinbjrnsdottir, E. Valdimarsson, F. Jakobsson, U. Agnarsson, V. Gudnason, G. Thorgeirsson, J. Fingerle, M. Gurney, D. Gudbjartsson, M. Frigge, A. Kong, K. Stefansson, and J. Gulcher. The gene encoding phosphodiesterase 4d confers risk of ischemic stroke. *Nature Genetics*, 35:131–138, 2003.

[41] A. J. F. Griffith, J. H. Miller, D. T.Suzuki, R. C. Lewontin, and W. M. Gelbart. *An introduction to genetic analysis.* W. H. Freeman, 7 edition, 2000.

[42] A. Gupta, J. Maňuch, L. Stacho, and X. Zhao. A characterization of haplotype inferring via galled-tree networks using a hypergraph covering problem. Accepted by Discrete Applied Mathematics.

[43] A. Gupta, J. Maňuch, L Stacho, and X. Zhao. Haplotype inferring via galled-tree networks is NP-complete. Accepted by COCOON 2008.

[44] A. Gupta, J. Maňuch, L. Stacho, and X. Zhao. Algorithm for haplotype inferring via galled-tree networks with simple galls (extended abstract). In *Proc. of International Symposium on Bioinformatics Research and Applications (ISBRA) 2007*, Lecture Notes in Bioinformatics, pages 121–132. Springer Berlin / Heidelberg, 2007.

[45] A. Gupta, J. v. d. Heuvel, J. Maňuch, L. Stacho, and X. Zhao. On the complexity of ordered colorings. Accepted by SIAM Journal on Discrete Mathematics, 2007.

[46] D. Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, 21:19 28, 1991.

[47] D. Gusfield. Inference of haplotypes from samples of diploid pupulations: complexity and algorithms. *Journal of Computational biology*, 8:305–323, 2001.

[48] 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.

[49] D. Gusfield. Haplotype inference by pure parsimony. Technical report, UC Davis Computer Science, 2003.

[50] D. Gusfield. Optimal, efficient reconstruction of root-unknown phylogenetic networks with constrained and structured recombination. *Journal of Computer and System Sciences*, 70:381–398, 2005.

[51] D. Gusfield and V. Bansal. A fundamental decomposition theory for phylogenetic networks and incompatible characters. In *Lecture Notes in Computer Science*, volume 3500, pages 217–232. Springer Berlin / Heidelberg, 2005.

[52] D. Gusfield, S. Eddhu, and C. Langley. Powerpoint slides for: Efficient reconstruction of phylogenetic networks (of SNPs) with constrained recombination. http://wwwcsif.cs.ucdavis.edu/~gusfield/talks.html.

[53] D. Gusfield, S. Eddhu, and C. Langley. The fine structure of galls in phylogenetic networks. *INFORMS Journal on Computing*, 16:459–469, 2004.

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

[55] E. Halperin and E. Eskin. Haplotype reconstruction from genotype data using imperfect phylogeny. *Bioinformatics*, 20:1842–1849, 2004.

[56] E. Halperin and R. M. Karp. Perfect phylogeny and haplotype assignment. In *RECOMB '04 Proceedings of the Eighth Annual International Conference on Research in Computational Molecular Biology*, pages 10–19, San Diego, California, USA, 2004. ACM press New York, NY. USA.

[57] International HapMap Project. http://www.hapmap.org/abouthapmap.html, 2002.

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

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

[60] L. Helmuth. Genome research: Map of the human genome 3.0. *Science*, 293:583 585, 2001.

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

[62] T. N. D. Huynh, J. Jansson, N. B. Nguyen, and W. Sung. Constructing a smallest refining galled phylogenetic network. In *RECOMB '05: Proceedings of the Ninth Annual International Conference on Research in Computational Molecular Biology*, Lecture Notes in Computer Science, pages 265–280. Springer Berlin / Heidelberg, 2005.

[63] National Human Genome Research Institute. Developing a Haplotype Map of the Human Genome for finding genes related to health and disease. Development Report, July 2001.

[64] The International HapMap Consortium. A haplotype map of the human genome. *Nature*, 437:1299 1320, 2005.

[65] J. Jansson, N. B. Nguyen, and W. Sung. Algorithms for combining rooted triplets into a galled phylogenetic network. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 349–358, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[66] J. Jansson, N. B. Nguyen, and W. Sung. Algorithms for combining rooted triplets into a galled phylogenetic network. *SIAM Journal on Computing*, 35:1098–1121, 2006.

[67] A. Jeffreys and L. Kauppi adn R. Neumann. Intensely punctate meiotic recombination in the class ii region of the major histocompatibility complex. *Nature Genetics*, 29:217–222, 2001.

[68] R. Judson and J. C. Stephens. Notes from the SNP vs haplotype front. *Pharmacogenomics*, 2:7–10, 2001.

[69] R. Judson, J. C. Stephens, and A. Windemuth. The predictive power of haplotypes in clinical response. *Pharmacogenomics*, 1:15–26, 2000.

[70] S. Kannan and T. Warnow. Inferring evolutionary history from DNA sequences. In *Proceedings of the 31st Annual Symposium on the Foundations of Computer Science*, pages 362–378, 1990.

[71] S. Kannan and T. Warnow. A fast algorithm for the ocmputation and enumeration of perfect phylogenies. *SIAM Journal on Computing*, 26:1749–1763, 1997.

[72] J. Z. Lin, A. Brown, and M. T. Clegg. Heterogeneous geographoic patterns of nucleotide sequence diversity between two alcohol dehydrogenase genes in wild barley (hordeum vulgare subspecies spontaneum). In *Proceedings of the National Academy of Sciences*, volume 98, pages 531–536, 2001.

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

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

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

[76] O. G. McDonald, E. Y. Krynetski, and W. E. Evans. Molecular haplotyping of genomic DNA for multiple single-nucleotide polymorphisms located kilobases apart using long-range polymerase chain reaction and intramolecular ligation. *Pharmacogenetics*, 12:93–99, 2002.

[77] F. R. McMorris, T. J. Warnow, and T. Wimer. Triangulating vertex colored graphs. *SIAM Journal on Discrete Mathematics*, pages 296 306, 1994.

[78] S. Michalatos-Beloin, S. A. Tishkoff, K. L. Bentley, K. K. Kidd, and G. Ruano. Molecular haplotyping of genetic markers 10 kp apart by allele-specific long-range PCR. *Nucleic Acids Research*, 24:4841–4843, 1996.

[79] 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 National Academy of Sciences of the United States of America*, volume 100, pages 5926–5931, 2003.

[80] L. Nakhleh, J. Sun, T. Warnow, C. R. Linder, B. M. E. Moret, and A. Tholse. Towards the development of computational tools for evaluating phylogenetic network reconstruction methods. *Pacific Symposium on Biocomputing*, 8:315–326, 2003.

[81] T. Niu. Algorithms for inferringn haplotypes. *Genetic Epidemiology*, 27:334–347, 2004.

[82] T. Niu, Z. S. Qin, X. Xu, and J. S. Liu. Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *The American Journal of Human Genetics*, 70:157–169, 2002.

[83] J. Odeberg, K. Holmberg, P. Eriksson, and M. Uhlen. Molecular haplotyping by pyrosequencing. *Biotechniques*, 33:1104–1108, 2002.

[84] C. H. Papadimitriou. *Computational Complexity*. Addison-Wisley Publishing Company, Inc., 1994.

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

[86] E. Pennisi. BREAKTHROUGH OF THE YEAR: Human Genetic Variation. *Science*, 318:1842 1843, 2007.

[87] D. Posada and K. Crandall. Intraspecific gene genealogies: trees grafting into networks. *Trends n Ecology and Evolution*, 16:37–45, 2001.

[88] G. Ruano and K. K. Kidd adn J. C. Stephens. Haplotype of multiple polymorphisms resolved by enzymatic amplification of single DNA molecules. *Proceedings of the National Academy of Sciences of the United States of America*, 87:6296 6300, 1990.

[89] G. Ruano and K. K. Kidd. Direct haplotyping of chromosomal segments from multiple heterozygotes via allele-specific PCR amplification. *Nucleic Acids Research*, 17:8392, 1989.

[90] R. Sachidanandam, D. Weissman, S. C. Schmidt, J. M. Kakol, L. D. Stein, G. Marth, S. Sherry, J. C. Mullikin, B. J. Mortimore, D. L. Willey, S. E. Hunt, C. G. Cole, P. C. Coggill, C. M. Rice, Z. Ning, J. Rogers, D. R. Bentley, P. Y. Kwok, E. R. Mardis, R. T. Yeh, B. Schultz, L. Cook, R. Davenport, M. Dante, L. Fulton, L. Hillier, R. H. Waterston, J. D. McPherson, B. Gilman, S. Schaffner, W. J. V. Etten, D. Reich, J. Higgins, M. J. Daly, B. Blumenstiel, J. Baldwin, N. Stange-Thomann, M. C. Zody, L. Linton, E. S. Lander, and D. Altshuler; The International SNP Map Working Group. A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. *Nature*, 409:928–933, 2001.

[91] A. J. Schafer and J. R. Hawkins. DNA variation and the future of human genetics. *Nature Biotechnology*, 16:33–39, 1998.

[92] M. H. Schierup and J. Hein. Consequences of recombination on traditional phylogenetic analysis. *Genetics*, 156:879–891, 2000.

[93] 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.

[94] Y. Song, Y. Wu, and D. Gusfield. Algorithms for imperfect phylogeny haplotyping with a single homoplasy or recombination event. In *WABI 2005*, Lecture Notes in Computer Science, pages 152–164, 2005.

[95] Y. S. Song. A concise necessary and sufficient condition for the existence of a galled-tree. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 3:160–186, 2006.

[96] S. Sridhar, G. E. Blelloch, R. Ravi, and R. Schwartz. Optimal imperfect phylogeny reconstruction and haplotyping (IPPH). In *Computational Systems Bioinformatics Conference Proceedings*, pages 199–210, 2006.

[97] M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9:91–116, 1992.

[98] U. Steidl, C. Steidl, A. Ebralidze, B. Chapuy, H. J. Han, B. Willl, F. Rosenbauer, A. Becker, K. Wagner, S. Koschmieder, S. Kobayashi, D. B. Costa, T. Schulz, K. B. O'Brien, R. G.W. Verhaak, R. Delwel, D. Haase, L. Trümper, J. Krauter, T. Kohwi-Shigematsu, F. Griesinger, and D. G. Tenen. A distal single nucleotide polymorphism alters long-range regulation of the PU.1 gene in acute myeloid leukemia. *The Journal of Clinical Investigation*, 117:2611 2620, 2007.

[99] J. C. Stephens, J. Rogers, and G. Ruano. Theoretical underpinning of the single-molecule-dilution (SMD) method of direct haplotype resolution. *The American Journal of Human Genetics*, 46:1149–1155, 1990.

[100] J. C. Stephens, J. A. Schneider, D. A. Tanguay, J. Choi, T. Acharya, S. E. Stanley, R. Jiang, C. J. Messer, A. Chew, J. Han, J. Duan, J. L. Carr, M.S. Lee, B. Koshy, A. M. Kumar, G. Zhang, W. R. Newell, A. Windemuth, C. Xu, T. S. Kalbfleisch, S. L. Shaner, K. Arnorld, V. Schulz, C. M. Drysdale K. Nandabalan, R. S. Judson,

G. Rua no, and G.F. Vovis. Haplotype variation and linkage disequilibrium in 313 human genes. *Science*, 293, 2001.

[101] M. Stephens and P. Donnelly. Inference in molecular population genetics. *Journal of the Royal Statistical Society. Series B.*, 62:605–655, 2000.

[102] M. Stephens and P. Donnelly. A comparison of bayesian methods for haplotype reconstruction from population genotype data. *American Journal of Human Genetics*, 73:1162 1169, 2003.

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

[104] E. A. Stewart, K. B. McKusick, A. Aggarwal, E. Bajorek, S. Brady, A. Chu, N. Fang, D. Hadley, M. Harris, S. Hussain, R. Lee, A. Maratukulam, K. O'Connor, S. Perkins, M. Piercy, F. Qin, T. Reif, C. Sanders, X. She, W. Sun, P. Tabar, S. Voyticky, S. Cowles, J. Fan, C. Mader, J. Quackenbush, R. M. Myers, and D. R. Cox. An STS-based radiation hybrid map of the Human Genome. *Genome Research*, 7:422–433, 1997.

[105] W. J. Strittmatter and A. D. Roses. Apolippoprotein E and Alzheimer's disease. *Annual Review of Neuroscience*, 19:53–77, 1996.

[106] G. A. Thorisson, A. V. Smith, L. Krishnan, and L.D Stein. The international hapmap project web site. *Genome Research*, 15:1591–1593, 2005.

[107] J. Tost, O. Brandt, D. Derbala, C. Caloustian, D. Lechner, and I. G. Gut. Molecular haplotyping at high throughput. *Nucleic Acids Research*, 30:e96, 2002.

[108] D. A. Tregouet, S. Escolano, L. Tiret, A. Mallet, and J. L. Golmard. A new algorithm for haplotype-based association analysis: the stochastic-EM algorithm. *Annals of Human Genetics*, 68:165–177, 2004.

[109] W. T. Tutte. An algorithm for determining whether a given binary matroid is graphic. In *Proceedings of the American Mathematical Society*, volume 11, pages 905–917, 1960.

[110] D. G. Wang, J. Fan, C. J. Siao, A. Berno, P. Young, R. Sapolsky, G. Ghandour, N. Perkins, E. Winchester, J. Spencer, L. Kruglyak, L. Stein, L. Hsie, T. Topaloglou, E. Hubbell, E. Robinson, M. Mittmann, M. S. Morris, N. Shen, D. Kilburn, J. Rioux, C. Nusbaum, S. Rozen, T. J. Hudson, R. Lipshutz, M. Chee, and E. S. Lander. Large-scale identification, mapping, and genotyping of single-nucleotide polymorphisms in the human genome. *Science*, 280:1077–1082, 1998.

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

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

[113] R. F. Weaver. *Molecular Biology.* the McGraw-Hill Companies, Inc., 2001.

[114] A. T. Woolley, C. Guillemette, C. L. Cheung, D. E. Housman, and C. M. Lieber. Direct haplotying of kilobase-size DNA using carbon nanotube probes. *Nature Biotechnology,* 18:760–763, 2000.

[115] Y. Wu and D. Gusfield. Efficient computation of minimum recombination with Genotypes (not Haplotypes). *Journal of Bioinformatics and Computational Biology,* 5:181–200, 2007.

[116] X. Zhong, P. M. Lizardi, X. Huang, P. L. Bray-Ward, and D. C. Ward. Visualization of oligonucleotide probes and point mutations in interphase nuclei and DNA fibers using rolling circle DNA amplification. *Proceedings of the National Academy of Sciences,* 98:3940–3945, 2001.