Using tree decomposition for general pedigree inference

by

Zhendong Sha

Bachelor of Science, Nanjing University Jinling College, 2013

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

In the Graduate Academic Unit of Faculty of Computer Science

Supervisor(s):	Patricia Evans, PhD, Computer Science
Examining Board:	David Bremner, PhD, Computer Science, Chair Michael Fleming, PhD, Computer Science Alyssa Sankey, PhD, Mathematics and Statistics

This thesis is accepted

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

December, 2018

©Zhendong Sha, 2019

Abstract

Genotype-phenotype linkage analysis is becoming more promising with recent advances in high-throughput sequencing technology and the HapMap project. Genetic investigation of heritability and SNP co-location works with haplotypes, single copies of genes; however, researchers often initially find genotype sequences, reflecting both copies of a gene, rather than haplotype sequences, partly because of economic and time concerns. As a result, an effective computational method of haplotype phasing, to infer haplotypes from genotypes, is needed.

The haplotype phasing problem can be classified into two subproblems concerning the existence of the relationship between individuals. This work focuses on individuals from a pedigree, and this phasing problem has been defined as the multi-recombinant haplotype configuration problem[24].

In this work, we propose a parameterized time algorithm, namely `DPTH`[1], to infer haplotypes from the genotypes of individuals related in a pedigree, using dynamic programming over a tree decomposition of the pedigree.

[1] `DPTH` stands for using **D**ynamic **P**rogramming over **T**ree decomposition of pedigree to find optimal **H**aplotype configuration for MRHC problem.

Dedication

To my parents, who dedicated their love and believe in me.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Patricia Evans, whose expertise, understanding, and patience, added considerably to my graduate experience. I appreciate her vast knowledge and skill in many areas (e.g., algorithm, bioinformatics), and her assistance in writing thesis.

I would also like to thank my family for the support they provided me through my entire life, without whose love, encouragement, I would not have finished this thesis.

Table of Contents

List of Figures

List of Symbols, Nomenclature or Abbreviations

MRHC	Multi-recombinant Haplotype Inference
DNA	Deoxyribonucleic acid
RNA	Ribonucleic acid
m	the number of sites in pedigree
n	the number of individuals in pedigree
SNP	Single Nucleotide Polymorphism
`DPTH`	Dynamic programming over tree decomposition haplotype inference
H	Haplotype enumeration set in bag
FPT	Fixed-Parameter Tractability
GWAS	Genome-wide associations studies

Chapter 1

Introduction

Linkages between deoxyribonucleic acid (DNA) mutation and genetic disease are extensively studied. With cheaper approaches to whole genome sequencing and Single Nucleotide Polymorphism (SNP) detection, analyses of SNPs in populations and between related individuals are increasingly common.

The majority of high-throughput sequencing technologies yield the genotype sequence, which is far more cost effective in comparison to the haplotype sequence. These two types of sequences can be viewed as the alternative representations of the genetic sequence, with haplotypes providing information about individual gene copies, while genotypes provide information about the pair together. Detailed introduction to these two types of sequences is discussed in Section 1.1.2.

Relationships between individuals are very informative in determining how disease related genes are inherited together, so this work focuses on trans-

lating genotype sequences to haplotype sequences, or haplotype phasing, in family pedigrees.

Mutation can be in the form of insertion, deletion, or repetition of nucleotides in DNA. The accumulation of mutations over the generations gives rise to the phenotype diversity of a population, which is encoded in the form of SNPs. An SNP[1] is a variation at a single position in a DNA sequence among individuals, if more than 1% of a population does not carry the same nucleotide in the DNA sequence [16]. There are millions of detected SNPs inside the human genome, with most of them considered harmless.

Recombinations[2] that happen during inheritance contribute significantly to genetic diversity as well as mutation, as they can combine different mutations. Considerable research is focused on locating harmful mutations, such as [31] [4], and determining how they are inherited. Genetic study requires haplotype sequences to understand inheritance patterns fully, but haplotype data is not always available in the initial phase of research [32]. Alternatively, genotype sequences, a compact form of haplotype sequences, are gathered instead. The problem of inferring haplotypes from genotypes, namely **haplotype inference** or **haplotype phasing**, thus arises. This work focuses on related individuals, using information about related individuals to infer haplotypes that minimize the number of recombinations in the pedigree. The

[1]For example, an SNP can change a DNA sequence from "ATCGTG" to "ATGGTG".

[2]The process of rearrangement of genetic material, especially by crossing over in chromosomes or by the artificial joining of segments of DNA from different organisms.

formal definition of the problem (Multi-recombinant[3] haplotype inference, MRHC) is given in chapter 1.2.

1.1 Biological prerequisites

Some biology prerequisites are essential for understanding the problem as well as the algorithm. This section will discuss the composition of the genome, its representation (haplotype and genotype sequence), the inheritance process of diploid organisms in a family pedigree, and the genome recombination event from a biological perspective.

1.1.1 Basics of genetics

Gregor Mendel observed the flowers of pea plants (diploid organisms) were either purple or white, but never an intermediate between the two colours. Mendel conceived the idea of heredity units, which he called "factors" and later hypothesized that allele pairs segregate from each other during the inheritance process. This is known as the Law of Independent Assortment or Mendel's Law [16].

The Boveri-Sutton chromosome theory, or the chromosome theory of inheritance, suggests chromosomes as the carrier of "factors", namely the gene [27], with chromosomes structured linearly. Later experiments proved that DNA is the genetic material that is responsible for inheritance [11].

[3]Recombinant is the result of the recombination event.

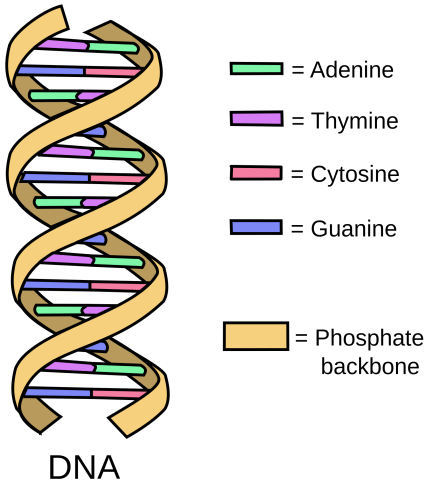


Figure 1.1: Example of DNA structure.

The discovery of the DNA double-helix structure took place in 1953 by James Watson and Francis Crick, using X-ray crystallography [26] (Figure 1.1). With the discovery of messenger RNA (mRNA), scientists learned how DNA controls protein production and in turn the inheritance of features [2].

The DNA of diploid organisms exists in pairs. The genome of humans consists of 23 pairs or 46 chromosomes in total. Each chromosome consists of millions of nucleotides, each of which is one of four nucleobases — adenine (A), cytosine (C), guanine (G), and thymine (T). The complete representation of DNA of a diploid organism is called its **haplotype sequence**, which consists of 2 strands over the alphabet $\{A, T, G, C\}$. Typical high throughput sequencing methods only yield information about the genotype [1], rather than haplotype. A **genotype sequence** represents the combination of the

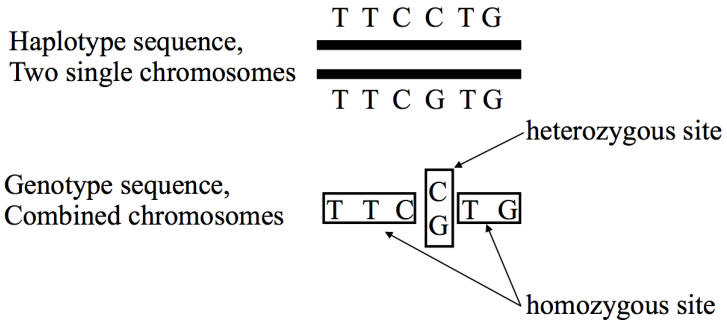


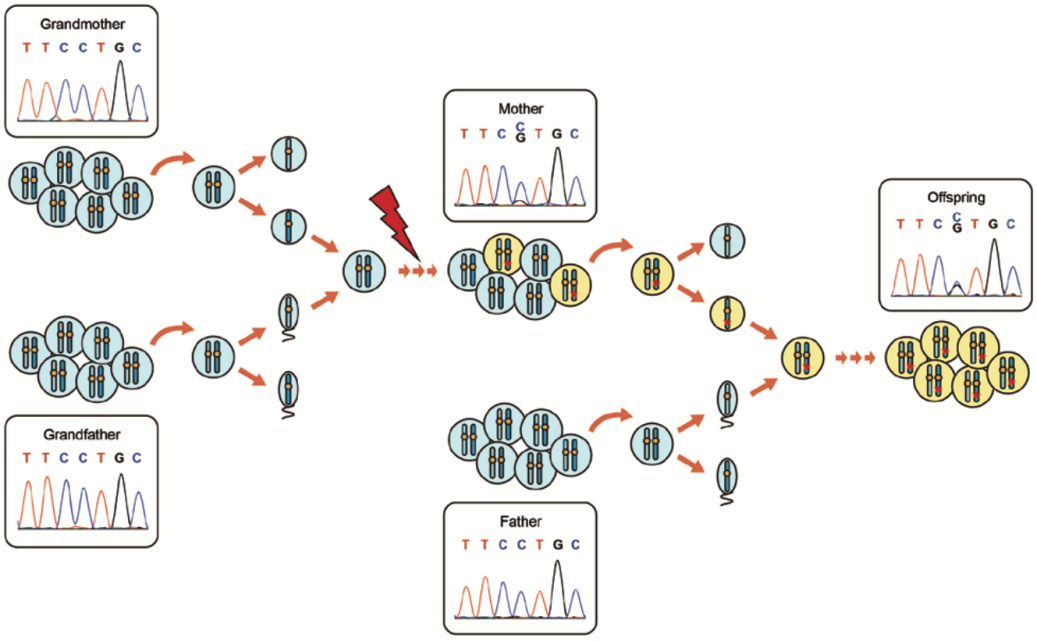
Figure 1.2: Example of haplotype and genotype.

bases of the haplotype sequences. If the bases of haplotype sequences at a position are the same on each strand, it is said to be a homozygous locus, and if they differ, it is said to be a heterozygous locus. An example of genotype sequence and its haplotype sequences pair is shown in Figure 1.2.

Loci or positions that have different values for some individuals than others are single nucleotide polymorphisms (SNPs), and often have only two different common variants [4].

Some real examples of a genotype sequence in a pedigree are shown in Figure 1.3 and [4], which show the genotyping of different individuals in the pedigree is only slightly different. In other words, heterozygous loci are rare.

Due to recent advances in high-throughput sequencing technology, there are extensive efforts to find mutations through genome-wide association studies (GWAS) [12] and mutation localization [31]. Pedigrees can further assist in investigating mutations and their inheritance in related individuals [18].



Cells in light yellow have heterozygous locus (introduced by the red light), cells in light blue do not have the heterozygous locus.

Figure 1.3: A real example [13] of pedigree with genotype sequence.

1.1.2 The formulation of haplotyping and genotyping

A haplotype sequence is abstracted from a sequence over $\{A, T, G, C\}$ to a vector over $\{0, 1\}$ [1], where 0 and 1 are used to represent the two common values at each locus. For a chromosome pair of a diploid organism with m loci, the haplotype sequence is denoted by a vector pair $\{h, h'\}$, where $h = \langle h_1, ..., h_m \rangle \in \{0, 1\}^m$ and $h' = \langle h'_1, ..., h'_m \rangle \in \{0, 1\}^m$.

The genotype sequence will also be transformed to the alphabet $\{0, 1, 2\}$, where 0 and 1 represent homozygous loci (both 0 or both 1), and 2 denotes a heterozygous locus (one 0 and one 1). Thus the genotype of a diploid organism is denoted by a vector $g = \langle g_1, ..., g_m \rangle \in \{0, 1, 2\}^m$.

The formulation of genotyping over $\{0, 1, 2\}$ rather than over pairs from $\{A, T, G, C\}$ can suffer from missing part of the detail, but in practice, it is still sufficient to describe the genotyping data and is commonly used in work on MRHC [19] [17] [7]. For haplotype sequences, SNPs from real datasets usually have two possible values, thus using $\{0, 1\}$ is enough to represent the different values at each locus. For genotyping sequences, the formulation should be able to describe the difference between homozygous loci $\{0, 1\}$ as well as heterozygous mutation $\{2\}$, so the alphabet $\{0, 1, 2\}$ is sufficient.

1.1.3 Pedigree

As mentioned previously, this work focuses on phasing haplotypes for individuals interconnected by a pedigree. The individuals in a pedigree (Figure

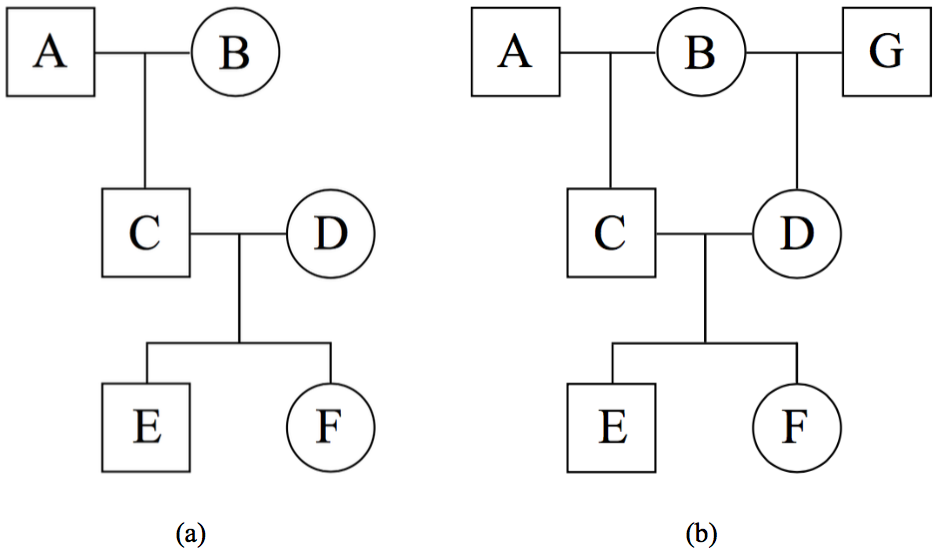


Figure 1.4: Example of pedigrees with (b) and without (a) mating loop.

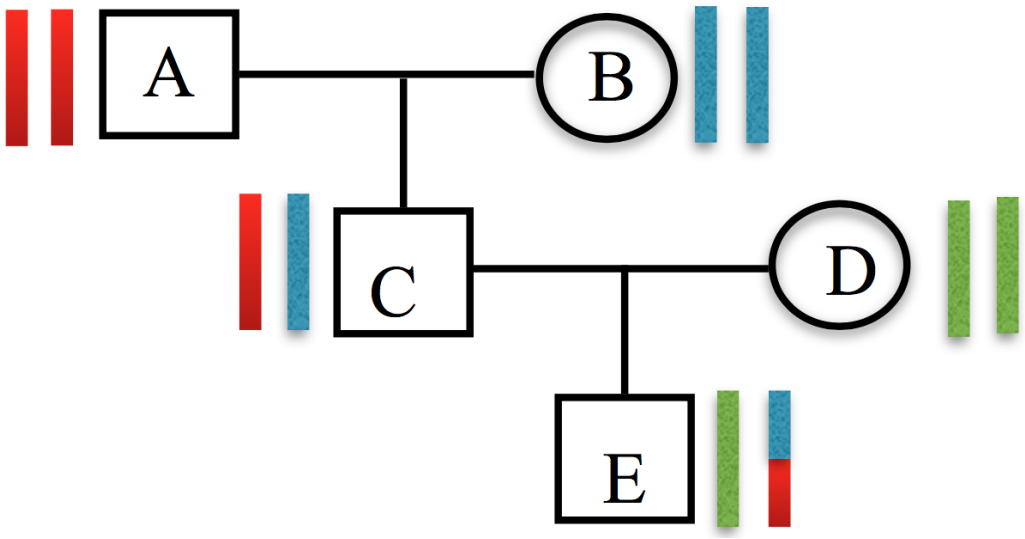
1.4) can be classified into two categories, male and female, based on gender. In the pedigree, a **family trio** consists of two parental members and a child individual. Thus, a pedigree can be viewed as the composition of family trios and independent individuals (the individuals that have neither parent nor child on record). A **mating loop** means for a family trio, the parental individuals have common ancestors in the pedigree. A general pedigree can have at least one mating loop. A tree pedigree has no mating loop. As shown in Figure 1.4, pedigree (a) is loop free, and pedigree (b) is a general pedigree with one mating loop, where individual $\{C\}$ and $\{D\}$ have common ancestor $\{B\}$.

1.1.4 Recombination event

Recombinants introduced by recombination events that happen during inheritance are a source of genetic diversity and propagation of harmful mutations. For diploid organisms, each chromosome (haplotyping sequence) has one copy from its father (paternal copy) and another from its mother (maternal copy). Recombination events will lead to a situation where the paternal copy is composed of two distinct grand-parental haplotype sequences. An example of recombination is in Figure 1.5, in which individual $\{E\}$ inherits a haplotype copy (with recombination) from paternal individual $\{C\}$, which is a composition of alleles from individual $\{A\}$ (in red) and individual $\{B\}$ (in green).

High-throughput sequencing technology gathers genotype sequences rather than haplotype sequences [1]. As shown in Figure 1.2, a haplotype sequence corresponds to one genotype sequence, but a genotype sequence can be mapped to at least one haplotype sequence, possibly more. This fact, suggested by Occam's razor[4] [22], gives rise to the need for a method to find the simplest hypothesis that fits the genotype data in a pedigree [15], which is considered to be the haplotype configuration with minimum recombinations in the pedigree.

[4]**Occam's razor:** Prefer the simplest hypothesis that fits the data.



Founder individuals' chromosome pairs denoted by bar pairs in different colours.

Figure 1.5: An example of a recombination event.

1.2 The Problem

The problem of inferring a haplotype configuration for individuals in a pedigree from genotype data is formally defined as Multi-recombinant haplotype configuration in pedigree (MRHC). The MRHC problem assumes there is no error in the genotypes of the pedigree. This is partly since introducing sequencing errors may increase the number of recombinations in the output configuration [24]. The MRHC problem is defined as follows.

MRHC:

Given: a pedigree P consisting of n individuals, and a genotype sequence $g_i \in \{0, 1, 2\}^m$ of length m for each member i.

Find: a pair of haplotype sequences $\{h_i, h'_i\}$, where $h_i, h'_i \in \{0, 1\}^m$ of length

m, for each member such that:

- haplotype pair h_i and h'_i are consistent with g_i;
- the transition of haplotype between generations follows the Mendelian law of inheritance with minimized recombination events;
- the number of recombinations is minimized.

This work will propose an algorithm, namely `DPTH`, to find an optimal solution to the MRHC problem by using dynamic programming over a tree decomposition of the pedigree.

1.3 Overview

This chapter discusses the fundamental biological definitions that are essential to understand the bioinformatics problem MRHC. Some review of the existing methods for the MRHC problem will be provided in Chapter 2.
The remainder of this thesis is organized as follows. Chapter 2 introduces the existing methods for MRHC, including special cases. Chapter 3 provides the technical details of our method. An execution example will be provided in Chapter 4. Chapter 5 proposes an extension of `DPTH`. Finally, Chapter 6 is the conclusion and future work.

Chapter 2

Background

The problem of MRHC is proved to be NP-hard by Li et al. [18]. Over the years, researchers have been exploring the MRHC problem and its special cases extensively.

The family of MRHC problems consists of four cases concerning the existence of loops and the recombination in the pedigree. We entitle the problem MRHC, assuming the pedigree may contain recombination events; or Zero-recombinant haplotype configuration (ZRHC) if the problems are assumed to have no recombination events. The MRHC problem and its special cases are summarized in Table 2.1. In this discovery, a pedigree will be decomposed into sub-pedigrees, using tree-decomposition.

Table 2.1: MRHC problem and its special cases.

		Recombination	
Mating loop		True	False
	True	MRHC	ZRHC
	False	MRHC on tree	ZRHC on tree

2.1 Related work

The relationships between genotype individuals can classify haplotype phasing problems into two categories: population data and pedigree data. This work addresses the problem of MRHC over pedigrees with mating loops and recombination events.

This section will highlight the obstacles that complicate the inference process, by reviewing the existing methods for all problem variants, and then introduce the essential ingredients of this work.

2.1.1 Existing methods for MRHC problem

Assuming that the pedigree is a general pedigree (with mating loops) with recombination events, Doan et al. proposed a graph theory based $O(2^k 2^{m^2} n^2 m^3)$ time algorithm [5], where n is the number of members, m is the number of sites, and k is the number of recombination events. This method formulated a chain of reductions eventually to the Bipartization by Edge Removal problem with additional parity-constraint sets S_{pc}.

Doan et al. also proposed a method for pedigrees with only two loci [6]. An iterative compression on a pedigree based graph is defined to produce

the time complexity $O(2^k n^2)$, where k is the parameter of the number of recombinations in the pedigree.

Based on Doan's work [7], Kirkpatrick introduced an algorithm that can find an optimal configuration in exponential time $O(n^{k+2}m^{6k})$ [14]. Kirkpatrick's method defined the minimum-recombination (MR) graph in addition to Doan's combinatorial method. The optimal solution to the MRHC problem is based on a colouring of the MR graph.

Before graph-based methods, many rule-based algorithms have been explored [10] [19] [18] [23] [24] [25]. For example, Li et al. proposed a heuristic algorithm [18] for the MRHC problem. It claimed to be very efficient when the hidden recombination event is rare. The underpinning assumptions of the heuristic method are:

- the expected number of recombination events within haplotype blocks is low;
- sibling haplotypes are significant evidence for inference;
- more than one recombination event is not likely to happen in one individual;
- individuals whose genotypes need a recombination event are advised to be phased last.

The time complexity of the heuristic algorithm is $O(n^2m^3)$. Since it is a heuristic method, it cannot guarantee to find an optimal assignment.

Assuming that the pedigree is a tree pedigree with recombination events, Xiao et al. presented a $O(mn \log^{k+1} n)$ time algorithm based on a stochastic model [30] that can find an optimal assignment with probability $(1 - O(k^2 \frac{\log^2 n}{mn} + \frac{1}{n^2}))$. The stochastic model is used to locate the recombination event; the pedigree will then be detached into recombination-free pieces and resolved independently.

2.1.2 Existing methods for ZRHC problem

A general pedigree without recombination events yields the zero-recombinant haplotype inference (ZRHC) problem. This assumption is motivated by the discovery of tightly linked markers in a small region [4], namely a haplotype block, in which a recombination event is an unlikely event [32].

Xiao et al. formulated the inheritance process into a system of linear constraints over the finite field of GF(2) and solved it in $O(mn^2+n^3 \log^2 n \log\log n)$ time [28]. Lai et al. proposed a linear-time $O(kmn + k^2m)$ algorithm that can find a general solution for the pedigree [17]. Lai's method was based on a spanning tree over the pedigree. It then generates inter-individual linear constraints for each locus respectively. The optimal solution is retrieved from the Gaussian elimination of the linear system. Doan et al. proposed a near-linear time $O(nm\alpha(m))$ algorithm [8] by setting up constraints between unresolved loci rather than individuals, where $\alpha(m)$ is the inverse of the Ackermann function.

Since there are no recombinations for ZRHC, recombinations are not a prob-

lem, but mating loops can give rise to inconsistencies. In Lai's work, haplotype inference is computed over a spanning tree, and some free variables are allowed in the resulting configuration, which corresponds to the fact that there may be multiple optimal configurations for a given pedigree.

Assuming that the pedigree is a tree pedigree without recombination events, Li et al. proposed an efficient algorithm using disjoint sets [20], which runs in a nearly linear time of $O(mn\alpha(n))$. Li's method solves the problem by formulating constraints as a linear system of variables. It then utilizes disjoint-set structures to encode connectivity information among individuals, to detect constraints from genotypes, as well as perform consistency checking of constraints. Liu et al. adapted Xiao's work [29] from general pedigrees to the simpler case of tree pedigrees [21]. Chan et al. introduced a linear time $O(mn)$ algorithm that finds one of the optimal haplotype assignments rather than a general solution [3].

2.1.3 Key issues of the MRHC problem

Mating loops and recombinations form the heart of the MRHC problem, which will complicate any optimization algorithm.

For the issue of recombinations, Lai et al. [17] and Doan et al. [8] proposed near-linear time algorithms based on linear programming and graph formulation respectively. Extending these methods to a more complicated case may be a good approach. But, when recombination happens, haplotype inference of these methods will be disrupted. In Lai's work [17], the linear constraint

system is inconsistent over a pedigree with recombinations.

For the issue of mating loops, some existing methods [17] [29] use a spanning tree to make the cross edge "invisible", in an adaptation of a tree pedigree solution. From our observation, the "invisible" edge may produce additional recombinants, since the corresponding 2 individuals exist in two distinct subtrees and are determined independently.

2.2 Tree decomposition

To handle the mating loop and the recombination event, `DPTH` will introduce tree decomposition and treewidth into the computation. This is because the family pedigree is a complex graph that tends to be somewhat tree-like, and so having a tree decomposition over the pedigree is a good approximation and can greatly narrow the search space.

A tree decomposition [9] of a graph $G = (V, E)$ is a tree τ together with a collection of subsets T_x (called bags) of V labeled by the vertices x of τ such that $\cup_{x \in \tau} T_x = V$ and the following connectivity properties hold:

1. (edge constraint) For every edge uv of G there is some x such that $\{u, v\} \subseteq T_x$.

2. (path constraint) If y is a vertex on the unique path in τ from x to z then $T_x \cap T_z \subseteq T_y$.

Treewidth w is the parameter of tree decomposition, which indicates the

complexity of the pedigree. The treewidth of a graph equals the maximum value of $|T_x|-1$ taken over all the vertices x of the tree τ of the decomposition. Larger treewidth will exponentially expand the search space of `DPTH`.

`DPTH` performs the search over the structure τ produced by tree decomposition using dynamic programming. Each bag of the tree decomposition narrows the search process to the local level. The structure that interconnects bags will collect fragments of the globally optimal solution from each local search.

2.3 Discussion

This chapter summarized the distilled core issues (mating loops and recombination events) of MRHC from the previous work. To handle loops and the resulting inconsistency issue, some previous work uses spanning trees and linear constraint systems. However, these linear systems are not able to process pedigrees with recombination events.

The emerging algorithm `DPTH` must handle loops and recombination events properly, so tree decomposition is introduced over the pedigree. Its promising tree-like structure over a complex graph can effectively assist `DPTH` to retrieve sufficient constraints, both local and global. Core concepts for `DPTH` are discussed in Section 3.7.

Chapter 3

Methodology

DPTH consists of the following steps. Firstly, DPTH reorganizes the pedigree using tree decomposition. Secondly, DPTH enumerates the potential optimal haplotype settings within each bag, from which the optimal assignment will be extracted. For each enumerated assignment, we define a score to indicate the quality (minimum number of recombinations). The optimal configuration will have the lowest score.

Recombinations are considered to be rare events, so we presume that they are not common in the pedigree, and do not occur densely in any part of the pedigree. As long as at least one of the family interactions for an individual does not involve recombination, we can restrict our search through their potential haplotypes as constrained by their family interactions. Only when these interactions conflict will we need to infer that a recombination is present, and consider only those possibilities that are implied directly by at

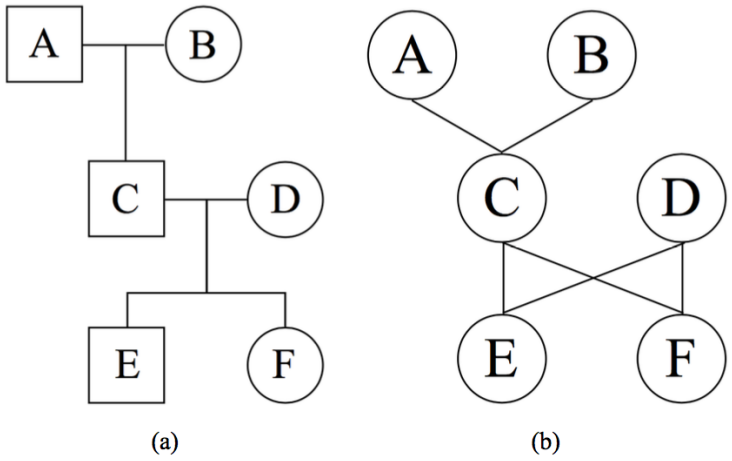


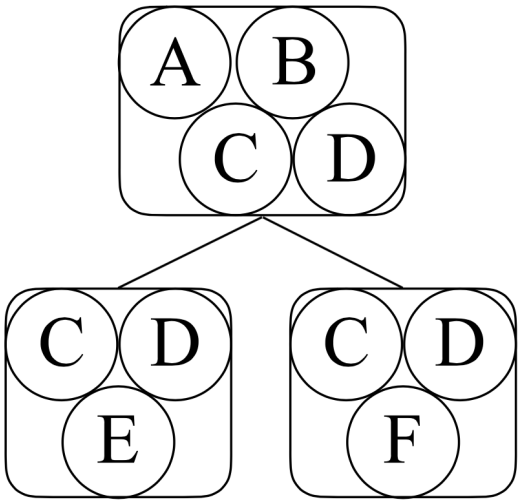
Figure 3.1: An example of a family pedigree and its corresponding graph.

least one interaction.

So, given a general pedigree with n individuals, m loci, and some recombination events, `DPTH` will perform a search over the space made from the reasonable enumeration of haplotypes (an exhaustive enumeration of recombinant-free assignments of each family trio), and finally reach an optimal haplotype configuration with minimum recombination events.

This approach is sufficient unless an individual has recombinations in all interactions. An extension, `DPTH-Ex`, is proposed in Chapter 5 to address this extreme situation by significantly expanding the search space at the cost of efficiency.

In this chapter, we will first discuss the fundamentals and important issues of `DPTH`; then the algorithm will be proposed step-by-step. Finally, the underlying insight will be discussed.



Circles represent the individual, squares and edges represent the structure of tree decomposition.

Figure 3.2: The tree decomposition for the graph in Figure 3.1.

3.1 Tree decomposition of pedigrees

Introducing tree decomposition into our method aims at reducing the search space, as well as parameterizing the time complexity of our method.

Because the pedigree is not a strictly defined graph, we have to slightly transform it into a graph. Each individual corresponds to a vertex in the graph. For each family trio, undirected edges should be set up between the paternal individual and the child as well as between the maternal individual and the child. An example of the pedigree-graph transformation is shown in Figure 3.1.

The corresponding graph $G = (V, E)$ of the pedigree is the combination of the vertex set V and edge set E. Since it originates from a pedigree, it en-

tails some special properties of the pedigree. Most pedigrees have tree-like structures (tree pedigree), but others have mating loops (general pedigree). The mating loop is one of the major obstacles that complicate the inference process, partly because the resulting multiple inference paths between individuals may give rise to inconsistencies and significantly expand the search space.

The tree decomposition of the graph in Figure 3.1 is shown in Figure 3.2. The complexity parameter **tree-width** of the tree decomposition does not need to be optimal, and our method will make sure each parent-offspring trio in the pedigree will only exist in one bag. Therefore the number of bags in the tree decomposition is the same as the number of parent-offspring trios in the pedigree data.

Each bag of the tree decomposition contains two kinds of individuals. The first part is the **anchor parent-offspring trio** (three individuals in total). Another part, namely **foreign individuals**, will make sure the two properties of tree decompositions, as discussed in Section 2.2, hold.

For the individual set in each bag, our method will define a set of haplotype assignments H and **scores** for each assignment in H . The score represents the number of recombinations, and it will be updated as the algorithm proceeds and eventually be used to find the optimal assignment.

3.2 Minimizing recombination

For any dynamic programming method, some optimal choices have to be made at each recursive step. `DPTH` is no exception. Considering the locally optimal choice only is not sufficient, because it can not guarantee a globally optimal assignment. To succeed, `DPTH` should accept some sub-optimal local assignments to achieve the optimal global assignment at the root.

Assuming there is no recombination event in the parent-offspring trio, then we can enumerate all reasonable haplotype assignments, as shown in the example of Figure 3.3. We refer to this exhaustive process as the **reasonable enumeration**.

This reasonable enumeration is inspired by the definition of "pre-determined loci", which has been defined in Lai's work [17]. Lai's work deems a pre-determined locus to be constant in the linear constraint system. This work, on the other hand, let these loci be constant during exhaustive enumeration and non-pre-determined loci be unknowns that need to be exhaustively enumerated.

Definition 3.2.1. *Pre-determined locus: For a parent-offspring trio, a pre-determined locus is a locus that is homozygous in the genotype sequence of at least one of the individuals in the trio.*

We introduce the notion of pre-determined locus partly because such a locus can be determined directly based on Mendel's law of inheritance. Any modification of such a locus (possibly in step 4 of `DPTH`) is equivalent to having

a recombination event in the current family trio.

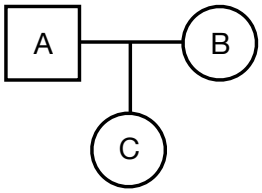
Definition 3.2.2. *Non-pre-determined locus: For a parent-offspring trio, a* Non-pre-determined *locus is a locus that is heterozygous in the genotype sequence of all three individuals in the trio.*

Theorem 3.2.1. *The number of reasonable enumerated configurations for a parent-offspring trio is $O(2^{\nu-1})$, where ν is the number of loci that are not pre-determined.*

Proof. During the enumeration process, the pre-determined loci are constant, and the non-predetermined loci will be enumerated. Each non-predetermined locus will be filled in from $\{0, 1\}$, so the enumeration set will expand to $O(2^{\nu})$. Considering the symmetrical assignments of a diploid organism, the size can be reduced by half to $O(2^{\nu-1})$. □

The reasonable enumeration process searches for the optimal configuration locally in each bag. Each bag will exchange configurations with its adjacent bags. We define this process as the enumeration set augmentation process, which will pave the road for the optimal configuration to show up at the end of dynamic programming.

Definition 3.2.3. *Enumeration set augmentation: Assume H_B is an assignment enumeration set for a bag B, individual i is a non-trio member in B, and h_i is a set of new assignments for an individual i of bag B. The augmentation process is $H_B \times h_i$, in which H_B will be augmented by entries of h_i.*



Genotypes:

	A	B	C
L1	1	2	2
L2	2	2	2
L3	2	2	2

Assignment set:

		A	B	C
Case 1	L1	1 1	0 1	1 0
	L2	0 1	1 0	0 1
	L3	0 1	1 0	0 1
Case 2	L1	1 1	0 1	1 0
	L2	0 1	1 0	0 1
	L3	1 0	0 1	1 0
Case 3	L1	1 1	0 1	1 0
	L2	1 0	0 1	1 0
	L3	0 1	1 0	0 1
Case 4	L1	1 1	0 1	1 0
	L2	1 0	0 1	1 0
	L3	1 0	0 1	1 0

The configuration in the reasonable enumeration is recombination free.

Figure 3.3: Example of reasonable enumeration of a parent-offspring trio.

The augmented set is the Cartesian product of the old set and assignments to augment. An example of this augmentation process is shown in Figure 3.5.

Throughout `DPTH`, the assignment set H in each bag will be initialized by the reasonable haplotype assignments of the anchor parent-offspring trio (step 2 of Algorithm), and then be augmented by copies of foreign individuals (step 3 of Algorithm) and anchor trio individuals (step 4 of Algorithm) in substructures.

This augmentation process will allow for haplotypes with recombinations occurring in the relationships of the bag, as long as there is some other relationship (bag) where those haplotypes do not involve recombinations.

For `DPTH`, a recombination event can be triggered in two ways: firstly, if the reasonable enumeration set is augmented by any new haplotype copies of trio individuals, then the resulting new assignment will have a higher score;

secondly, the difference between the root assignment set H and the H sets in substructures would also increase the score at the root through the optimal choices made by the scoring function. `DPTH` defines a variable w for each entry in the enumeration set to indicate the score, or quality, of each assignment. Each indicator w will be initialized to be 0 in step 2, updated in step 4, and evaluated by the scoring function in Section 3.3.

3.3 Scoring function

A recursive scoring function will be performed over the tree decomposition to evaluate the quality of each entry in all H sets from an inter-bag or global perspective. The recursive function is as follows:

$$r(h_R) = \sum_{C_i \in C} \min_{h_j \in C_i} \{r(h_j) + d(h_j, h_R)\} \tag{3.1}$$

where h_R represents one of the assignments in enumeration set H at root R, C represents substructures rooted at bag R, C_i represents the set of roots of substructures, h_j represents one of the haplotype assignments in H of C_i, and the linear time function $d(h_j, h_R)$ calculates the minimum number of recombinations needed to transform h_j to h_R. The pseudocode of linear time function d is in Algorithm 4.

3.4 The mechanism of DPTH

DPTH enumerates the recombination-free assignments in each bag and finds the optimal configuration from the exhaustive enumeration of all possible combinations of copies that exist in the substructure using the scoring function.

A visualized high-level introduction to the methodology of DPTH is provided in Figure 3.4. The configuration space represents the enumeration set of each bag, which contains different configurations and their corresponding weights. For each pair of entries in the space, there exists a distance that represents the number of recombinations needed to transform one to the other. A dashed line between two entries denotes interactions of entries. The inference process of DPTH is to find the entry with the optimal recombination number (indicated by weight) by considering the distance between entries and the weight of entries.

As shown in Figure 3.4(a), the configuration space of the root bag (Bag 1) and substructure (Bag 2) have overlapping sub-space, so the entry a in the overlapping space will not suffer from distance penalty, and configurations in such a sub-space are very likely to be the optimal overall assignment. Note that, in order to tolerate sub-optimal assignments, the weight of all entries will be considered, and this makes DPTH behave in an enumerative manner. Due to DPTH initializing and augmenting the configuration space of each bag separately, it is possible that the configuration spaces of two adjacent bags

do not have any overlap. As shown in Figure 3.4(b), each of the entries in the root bag (Bag 1) have to find the optimal choice by considering both the distance measure and the weight of entries in Bag 2.

Finally, at the root of the entire tree decomposition, `DPTH` selects the entry with the lowest weight to be the optimal configuration. Figure 3.4(c) highlights two typical entries at the root bag (Bag 1), a and b. If `DPTH` decides a is the optimal configuration, then `DPTH` thinks no recombination is needed to better accommodate the substructure. If `DPTH` decides b is the optimal entry at the root bag, then `DPTH` thinks "paying" the distance between b and c is the best "deal".

3.5 Algorithm

The algorithm consists of 6 steps. In step 1, a tree decomposition is generated based on the pedigree; step 2 initializes the haplotype assignment set H of all bags with the enumeration of all reasonable haplotype settings of the anchor trio; in step 3, the enumerated haplotype set H is augmented by the copies of foreign individuals existing in the substructure; in step 4, additional copies of trio individuals in the substructure are augmented into assignment set H as well as used to update the score of that haplotype setting; in step 5, each haplotype setting is evaluated by the recursive score function; finally in step 6, one of the optimal haplotype assignments of this pedigree is fetched from the enumeration set in each bag recursively.

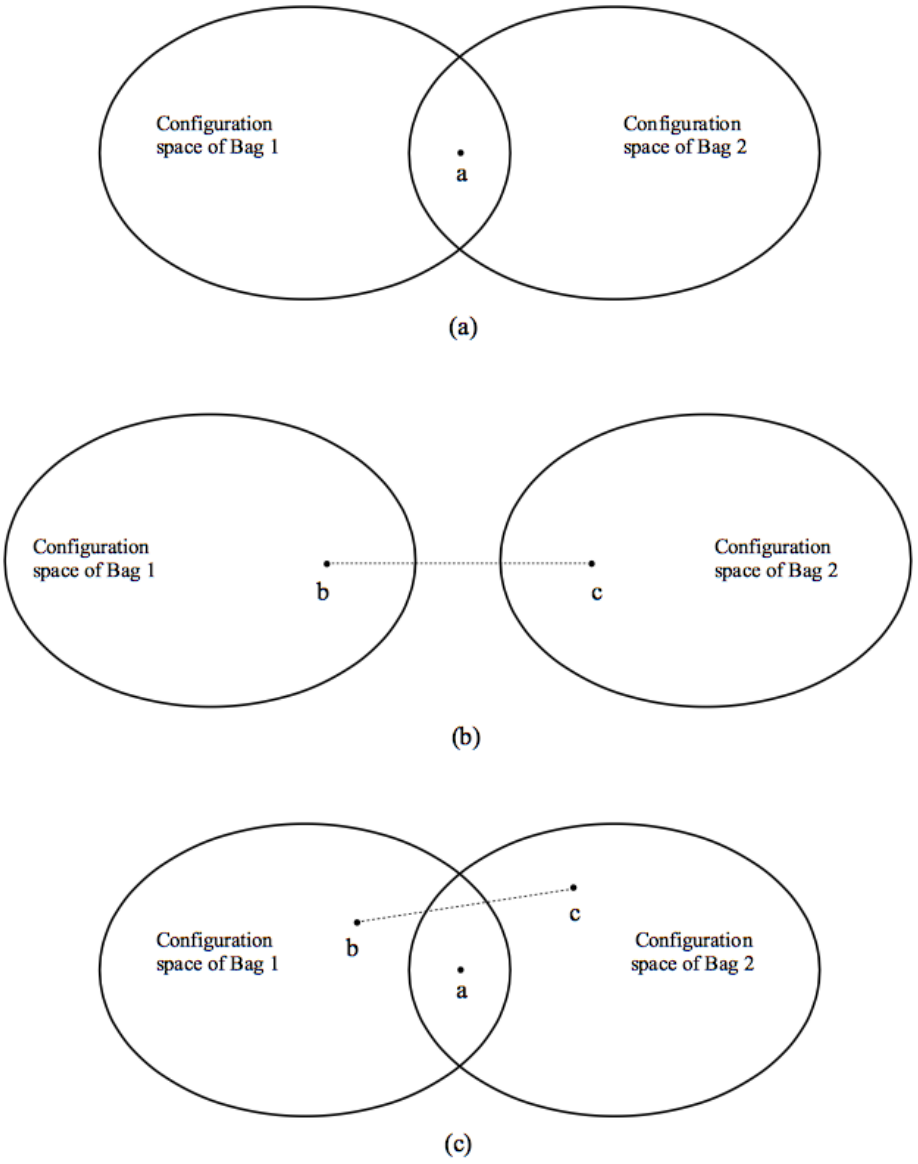


Figure 3.4: Visualized high-level introduction to the methodology of DPTH.

Step 1: preprocessing

In this step, a tree decomposition of the given pedigree is generated. The tree decomposition does not need to be optimal in tree-width, and more complex techniques that produce decomposition of smaller width can potentially be adapted to be used here.

One may ask why using an optimal tree-width is not essential. The reasons are as follows: firstly, finding an optimal tree-width is a difficult task; secondly, the bags of the tree decomposition are made up of family trios, so the tree decomposition may not be optimal; thirdly, actual pedigrees, especially for human beings, do not contain many mating loops.

The procedure is as follows: The anchor parent-offspring trio of each bag is first initialized to be a single parent-offspring trio in the pedigree. After that, our method determines the potential edges (edges between bags that have individuals in common) of the tree decomposition by a pairwise comparison of bags and then sets up the structure of tree decomposition using breadth first search and initializes the **foreign individuals** in each bag by backtracking. If there is an individual in two or more distinct substructures, then backtracking is triggered to insert the individual into the bags along the path in between.

Theorem 3.5.1. *Step 1 (preprocessing) of* `DPTH` *is* $O(n^2)$ *time, where* n *is the number of individuals.*

Proof. Initializing a bag for each parent-offspring trio in the pedigree is $O(n)$ time. To determine the candidates of the edge in a tree decomposition can

be done in time $O(n^2)$, as there are at most n bags to compare and each comparison (detecting the common individuals) takes constant time. The breadth first search takes $O(n)$ time and each backtracking takes $O(n)$ time (at most n calls). □

Step 2: initialize the assignment set H

In this step, the H set in each bag will be initialized to be the reasonable enumeration of recombination free assignments. As mentioned in Section 3.2, reasonable enumeration is based on the definition of pre-determined locus and non-pre-determined locus. During this enumeration process, pre-determined loci are treated as constant, and non-pre-determined loci are treated as variables over $\{0, 1\}$ that are to be enumerated.

The enumeration process is as follows. Firstly, the pre-determined loci will first be filled in with respect to Mendel's law of inheritance; secondly, the non-pre-determined loci of child and paternal individuals will be enumerated simultaneously with the same value; lastly, the non-pre-determined locus of the maternal individual will also be filled in accordingly, but with opposite value to the child. For example, when the child locus is $\langle\langle 0\rangle, \langle 1\rangle\rangle$, then the maternal locus is $\langle\langle 1\rangle, \langle 0\rangle\rangle$. And when the child locus is $\langle\langle 1\rangle, \langle 0\rangle\rangle$, then maternal locus is $\langle\langle 0\rangle, \langle 1\rangle\rangle$

We define H set to contain all recombinant-free assignments of the parent-offspring trio. Enumeration over all non-predetermined loci is performed, which means that the size of H at the end of this step is 2^x, where x denotes

the number of non-predetermined loci [1] . A small example of trio enumeration is showed in Figure 3.3.

Theorem 3.5.2. *Step 2 of* `DPTH` *is* $O(n2^m)$ *time, where* n *is the number of indiviuals and* m *is the number of loci.*

Proof. The worst case is given all the loci in the parent-offspring trio are non-predetermined. According to Theorem 3.2.1, there are at most $O(2^m)$ cases enumerated in the worst case, where the number of non-predetermined loci ν equals the number of sites of individual m. So initializing H in all $O(n)$ bags should, therefore, take $O(n2^m)$ time. □

Step 3: augment assignment set H with copies of foreign individuals in substructure

In this step, our method recursively augments the enumeration set H with copies of foreign individuals in the substructure. For each foreign individual i our method first fetches all copies $Copy_i$ from the substructures, and then H is augmented accordingly. The pseudocode is in Algorithm 1 .

This augmentation process is also performed by enumeration, and the new H is the Cartesian product of the "old" H and $Copy_i$. As illustrated in the example in Fig 3.5, all copies in $Copy_i$ are augmented into H . If $|Copy_i| = s$ and $|H| = h$, then the final size of H is $O(hs)$.

[1] In real genotype data, the heterozygous locus is rare [4], so the number of non-predetermined loci (needs three individual in the trio to be heterozygous at the same locus) should be even smaller.

Theorem 3.5.3. *There are at most $w-2$ foreign individuals in a bag, where w is the treewidth of the tree decomposition.*

Proof. The treewidth of a tree decomposition of a pedigree is w, so there are at most $w+1$ individuals, or vertices, in each bag. Knowing that three individuals are anchor parent-offspring individuals and individuals in a bag consist of anchor parent-offspring trio and foreign individuals, there are at most $w+1-3 = w-2$ foreign individuals in the biggest bag in size. □

Theorem 3.5.4. *There are at most $2^{\Delta-1}$ haplotype sequence configurations for an individual with Δ heterozygous loci.*

Proof. Each heterozygous locus has two possible values $\{0, 1\}$, so there are 2^{Δ} possible combinations of assignments. Considering the symmetrical assignments to be equal, there are at most $\frac{2^{\Delta}}{2} = 2^{\Delta-1}$ haplotype entries for a individual with Δ heterozygous loci. □

Theorem 3.5.5. *Step 3 of* `DPTH` *is $O(n2^{mw-m})$ time, where w is the treewith of the tree decomposition, n is the number of individuals and m is the number of loci.*

Proof. The size of H before step 3 is $|H|$ and the size of the largest $Copy_i$ of each foreign individual is Γ. The Cartesian product of each augmentation of $Copy_i$ will grow set H to $|H| \times \Gamma$. As there are at most $O(w-2)$ foreign individuals, the size of H is $|H|\Gamma^{w-2}$. Knowing that H after step 2 is capped at $O(2^m)$ ($|H| = O(2^m)$), $\Gamma \leqslant 2^m$, and there are at most n bags, after

applying step 3, the summation of the size of all enumeration sets H is $O(n2^{mw-m})$. □

Algorithm 1: InitForiIndi

Input : tree decompsition *td*

Output: initialized foreign individuals of each bag in *td*

```
1  begin
2      postOrderTrav ← PostOrderTraversal (td)
3      foreach bag p_i in postOrderTrav do
4          children← substructures rooted at p_i
5          if children.size = 0 then
6              continue
7          foreach foreign individual indi_j in bag p_i do
8              allCopy ← {}
9              foreach child bag c_k in children do
10                 copies ← get all copies of indi_j in substructure rooted at
                   c_k
11                 allCopy ← allCopy ∪ copies
12             Augment the enumeration set of p_i with allCopy of indi_j
13     return
```

Step 4: Augment H set with multiple copies of trio individuals

For each individual of the anchor parent-offspring trio, there may be new copies in substructures. To take these copies into consideration, assignment

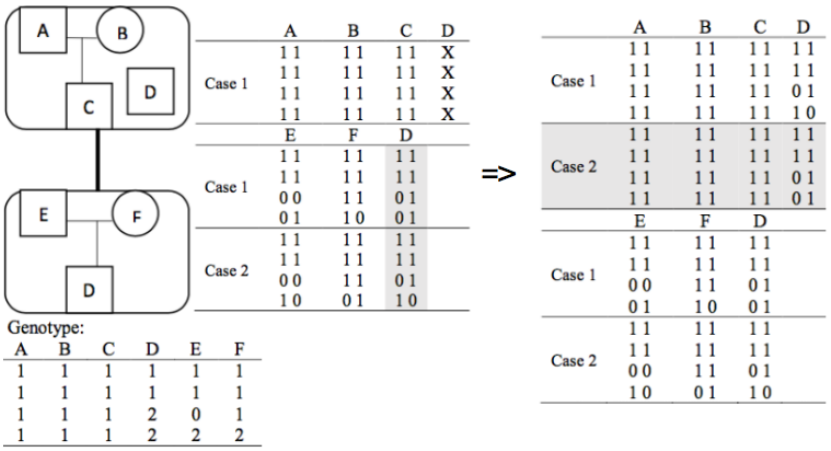


Figure 3.5: Example of augmentation of set H with copies of foreign individual.

set H is augmented, and the scores are updated accordingly.

Tolerating locally suboptimal assignments (containing recombination events) is to trade a score increase at the root bag for a decrease in the overall score. A small example of step 4 is shown in Figure 3.6 and Figure 3.7. As there is no "compromise" made in Figure 3.6, the only haplotype assignment at the root has to suffer from two "collisions" between root assignment set H and the substructure assignments. Due to step 4, a new configuration is augmented into the H set at root bag $\{A, B, C\}$. This new assignment looks "unreasonable" for the root bag, but the sacrifice at the root is essential for reaching an optimal assignment (H set after step 4 is in Figure 3.7).

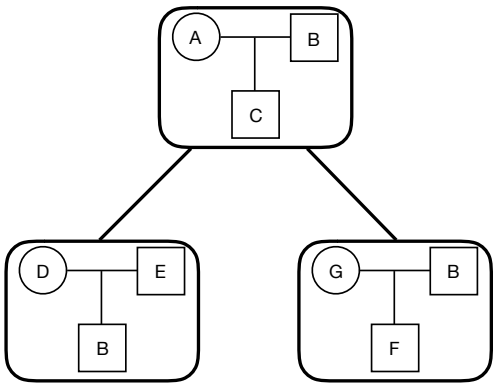
Algorithm 2: RecursiveLabelMultiCopyTrio

Input : tree decomposition *td*, the root bag r of *td*

Output: updated copy of each trio member and value r

```
1  begin
2      if r is the leaf in td then
3          return
4      else
5          children ← substructures rooted at r
6          foreach bag b_i of children do
7              RecursiveLabelMultiCopyTrio(b_i)
8          foreach bag p_i of children do
9              copyset ← {}
10             if r.father resolved in p_i then
11                 copyset ← get all copies of r.father in substructure p_i
12             if r.mother resolved in p_i then
13                 copyset ← get all copies of r.mother in substructure p_i
14             if r.child resolved in p_i then
15                 copyset ← get all copies of r.child in substructure p_i
16             Augment the enumeration set H of r with copyset
17     return
```

To achieve this effect, our method first fetches new copies from the substructure and then augments H set using enumeration. In other words, the new



	A	B	C
Case 1 w=0	11	10	10
	11	10	10
	11	10	10

	D	E	B
Case 1 w=0	00	11	10
	00	11	10
	11	00	01

Genotypes:

A	B	C	D	E	F	G
1	2	2	0	1	2	1
1	2	2	0	1	2	1
1	2	2	1	0	2	0

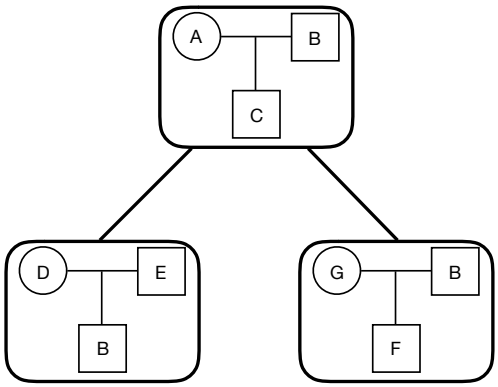
	B	G	F
Case 1 w=0	01	11	10
	01	11	10
	10	00	01

Figure 3.6: Set H of tree decomposition before step 4.

H after this augmentation process is the Cartesian product of "old" H and the new copies. The pseudocode in this step is specified in Algorithm 2. The size that the new H will increase to in this step depends on the number of new instances that each of the trio individuals has in substructures.

Theorem 3.5.6. *Step 4 of* `DPTH` *is* $O(n2^{2m+mw})$ *time, where* w *is the tree-with of the tree decomposition,* n *is the number of individuals and* m *is the number of loci.*

Proof. Assume that the size of H is $|H|$ after step 3. For each anchor individual, there are $O(2^m)$ new instances in the substructure. So each cartesian product augmentation process will at most grow H from $|H|$ to $|H| \times 2^m$. Since there are exactly three anchor parent-offspring individuals, the size



	A	B	C
Case 1 w=0	11	10	10
	11	10	10
	11	10	10
Case 2 w=1	11	10	10
	11	10	10
	11	01	10

	D	E	B
Case 1 w=0	00	11	10
	00	11	10
	11	00	01

	B	G	F
Case 1 w=0	01	11	10
	01	11	10
	10	00	01

Genotypes:

A	B	C	D	E	F	G
1	2	2	0	1	2	1
1	2	2	0	1	2	1
1	2	2	1	0	2	0

Figure 3.7: Set H of tree decomposition after step 4.

of H after step 4 is $|H| \times 2^{3m}$. Knowing that the size of H after step 3 is $O(2^{m+m(w-2)})$, the size of H after step 4 is $O(2^{m+m(w-2)} \times 2^{3m})$. Since there are $O(n)$ bags, the time complexity of step 4 is $O(n2^{m+m(w-2)+3m}) = O(n2^{2m+mw})$. □

Step 5: scoring each haplotype setting in each bag

In this step, the method scores all haplotype assignments in all bags. The scoring process is based on the recursive scoring function (3.1), which is used to compare each assignment set H at the root with each H set in substructures. `DPTH` compares each assignment at the root bag with assignments in the substructure. The scoring process is performed over the tree decomposition in a recursive manner, which is specified in Algorithm 3.

Theorem 3.5.7. *Step 5 of* DPTH *is* $O(mn^2 2^{2mw+4m})$ *time, where* w *is the tree-with of the tree decomposition,* n *is the number of individuals and* m *is the number of loci.*

Proof. Assuming that the maximum H size that is involved in this recursion is $|H|$, then in the overall scoring process $O(|H|^2)$ comparisons are needed, where each comparison takes $O(mn)$ time. After step 4, the size of H is $O(2^{m+m(w-2)+3m})$. Therefore each recursion takes $O(mn2^{2(m+m(w-2)+3m)})$ time. Over all $O(n)$ bags, the scoring process takes $O(mn^2 2^{2(m+m(w-2)+3m)}) = O(mn^2 2^{2mw+4m})$. □

Algorithm 3: RecursiveScoreValueR

Input : tree decomposition td, the $root$ bag of td

Output: update all $value_R$ at bag $root$

```
1  begin
2      if root is leaf in td then
3          return
4      else
5          children ← substructures rooted at root
6          foreach root bag c_i in children do
7              RecursiveScoreValueR(c_i)
8          foreach assignment h_i in H set of root bag do
9              h_i.value_R ← h_i.value_R +
                ∑_{C_i ∈ children} min_{h_j ∈ C_i} {r(h_j) + d(h_i, h_j)}
10     return
```

Step 6: fetching the optimal assignment

In this final step, an optimal configuration of haplotypes for the input pedigree is fetched recursively. After step 5, the optimal solution is now hidden in the H set and the scoring array of each bag. After performing Algorithm 5, an optimal solution is finally formalized. In Algorithm 5, for each assignment in H at the root, the method finds one assignment that is the optimal match from each substructure and merges them.

Theorem 3.5.8. *Step 6 of* `DPTH` *is* $O(mn^2 2^{2mw+4m})$ *time, where* w *is the*

Algorithm 4: Function d

Input : two copies, *root* and *child*, of a individual, the genotype g of the individual

Output: the minimum number of recombinations needed between 2 copies

```
1  begin
2      foreach site s_i of g do
3          array[s_i] ← root[s_i] ⊕ child[s_i]
4      heteArray ← array
5      foreach site s_i of g do
6          if g[s_i] is homozygous then
7              delete s_i from heteArray
8          else
9              continue
10     counter ← 0
11     foreach slot s_i of heteArray do
12         if slot s_i differ from s_i − 1 then
13             counter ← counter + 1
14         else
15             continue
16     return counter
```

tree-with of the tree decomposition, n is the number of individuals and m is the number of loci.

Proof. Assume the maximum set H size is $|H|$. Finding the optimal match by $O(mn)$ time in a two-by-two comparison involves $O(|H|^2)$ comparisons, where $|H| = O(2^{m+m(w-2)+3m})$. As there are $O(n)$ bags, this step takes $O(mn^2 2^{2(m+m(w-2)+3m)})$ or $O(mn^2 2^{2mw+4m})$ time. □

Algorithm 5: FetchOptConfiguration

```
   Input  : tree decomposition td, the root bag of td
   Output: Optimal haplotype configuration
 1 begin
 2     if root is leaf in td then
 3         return
 4     else
 5         children ← substructures rooted at root
 6         foreach bags c_i in children do
 7             FetchOptConfiguration(td, c_i)
 8         foreach assignment h_i in H set of root bag do
 9             foreach c_i in children do
10                 Find a configuration c in c_i that has minimum difference
                    to h_i
11                 Augment h_i with c
12     return
```

3.6 Time complexity of DPTH

Theorem 3.6.1. *The time complexity of* DPTH *is $O(mn^2 2^{2mw+4m})$, where w is the tree-with of the tree decomposition, n is the number of individuals and*

m is the number of loci.

Proof. DPTH consists of six steps, and the overall time complexity is the sum. As mentioned, step one takes $O(n^2)$ time; step two takes $O(n2^m)$ time; step three takes $O(n2^{m+m(w-2)})$ time; step four takes $O(n2^{m+m(w-2)+3m})$ time; step five takes $O(mn^2 2^{2mw+4m})$ time; step six takes $O(mn^2 2^{2mw+4m})$ time. So the overall time complexity of DPTH is $O(mn^2 2^{2mw+4m})$. □

DPTH is a parameterized time algorithm with two parameters, which are w – the tree-width, and m – the number of loci. The parameter w indicates the complexity of the pedigree, which is mostly contributed by the mating loops in the pedigree. In this work, DPTH initializes the structure of tree decomposition (step 1) by parent-offspring trio based bags and breadth first search. This approach is non-optimal and may yield large w in a worst case scenario. However, we deem this approach is sufficient here, as large w will only boost time and space requirements but will not disrupt enumeration and in turn the inference process. Finding tree decomposition of low treewidth is itself a significant research problem [9].

The parameter m in the overall time complexity is the maximum number of non-predetermined loci, which is believed to be very small in real pedigree genotype data. An example genotype pedigree in [4] reveals that heterozygous loci are very rare in a pedigree, which gives us the confidence to assume the non-predetermined locus is even more rare, as a non-predetermined locus in a family trio must have three heterozygous loci in three individuals at the

same locus.

3.7 Discussion

To sum up, DPTH performs the search based upon the tree decomposition over a pedigree. The bags of the tree decomposition split and narrow the overall search space into sub-spaces, and the sub-spaces will synchronize with the adjacent bags to make sure the globally optimal configuration is reached. We put the search within a bag as a local perspective, and put the search across bags as the global perspective.

In theory, the potential search space is enormous. Searching for the optimal configuration from thousands of non-optimal settings in the pedigree is the same as finding the lost pear in tons of potatoes. The global perspective of DPTH applies dynamic programming on the tree decomposition, which will maximally utilize the local search. Thus the DPTH has smaller search space than brute force.

DPTH omits a special case of recombination to trade for efficiency. DPTH assumes a paternal individual will not pass a haplotype with the same recombination to all its children (where the number of children is more than one). If an individual passes down the allele with recombination to all children, then the optimal assignment will never be enumerated by DPTH. DPTH-Ex, an extension of DPTH, is defined for this special case.

3.7.1 The local perspective of DPTH

The local perspective of DPTH consists of step 2 and step 3. The local perspective will enumerate all zero recombination configurations of the host family trio of each bag. Moreover step 3 will fill-in the foreign individuals of each bag using copies that exist below. The number of foreign individuals in the bag will affect the scope of local perspective.

3.7.2 The global perspective of DPTH

The global perspective will perform the search by introducing recombination events. Step 4 will interfere the reasonable enumeration of a family trio with reasonable enumerations from other trios. And step 5 will evaluate the configuration set using the scoring function, which will also increase the recombination number.

For an individual, there may be different configurations in different areas of the pedigree. The global perspective will find a configuration that best matches all trios.

Chapter 4

Illustrative Example

The easiest way to illustrate the algorithm is via an example.

4.1 An execution example

In this section, we will perform our method step by step over a tiny pedigree (Figure 4.1(a)). As we are addressing the general MRHC problem, the pedigree consists of one mating loop and one recombination event. The corresponding tree decomposition of the pedigree is in Figure 4.1(b) (as generated by `DPTH` in the preprocessing step), and the recurrence is rooted at bag $\{C, E, F, B\}$.

In the second step, the method initializes the set H of each bag with "reasonable" haplotype configurations of the anchor parent-offspring trio. As is shown in Figure 4.2, the score of each assignment is 0, as they are all "reason-

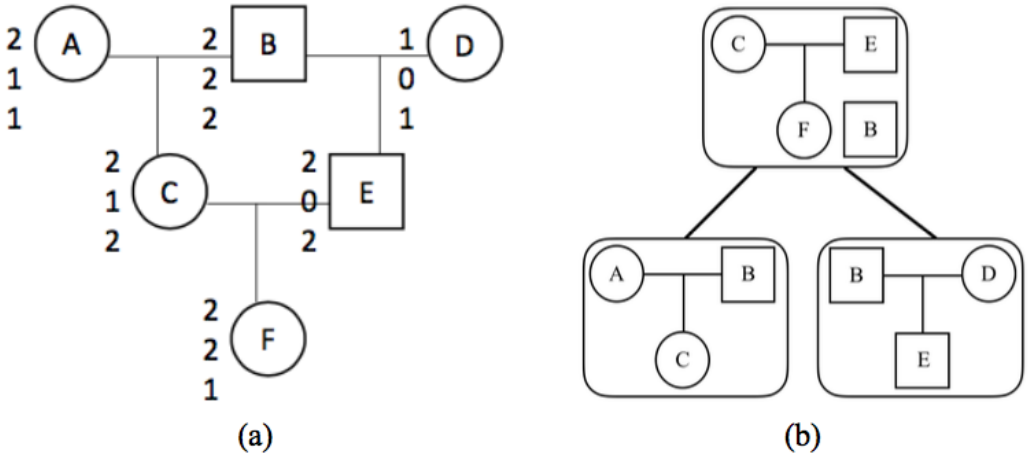


Figure 4.1: The example pedigree and the corresponding tree decomposition.

able" and the size of H depends on the number of non-predetermined loci in each family. For example, parent-offspring trio $\{C, E, F\}$ of bag $\{C, E, F, B\}$ has only one non-predetermined locus, so the resulting H has 2 assignments.

In the third step, all H are augmented by copies of foreign individuals in substructures. In this example, only bag $\{C, E, F, B\}$ has non-trio individual $\{B\}$. There are three distinct haplotype copies in substructures, and the H is increased to six assignments (the expanded H set is in Figure 4.3).

In the fourth step, all H are augmented by copies of trio individuals in substructures by enumeration. In this example, individual $\{C\}$ and $\{E\}$ of bag $\{C, E, F, B\}$ have more copies in substructures. By adding copies into the H of bag $\{C, E, F, B\}$, the size of H is increased, and the score of the new assignment is increased concerning the recombination that is needed between the new assignment and the original one. After the augmentation process,

Reasonable enumeration (step 2)

	C	E	F	B
Case 1 w = 0	1 0 1 1 1 0	0 1 0 0 1 0	1 0 1 0 1 1	X X X
Case 2 w = 0	0 1 1 1 1 0	1 0 0 0 1 0	0 1 1 0 1 1	X X X
	A	B	C	
Case 1 w = 0	0 1 1 1 1 1	1 0 1 0 0 1	0 1 1 1 1 0	
Case 2 w = 0	0 1 1 1 1 1	0 1 1 0 0 1	1 0 1 1 1 0	
	B	D	E	
Case 1 w = 0	0 1 0 1 0 1	1 1 0 0 1 1	1 0 0 0 1 0	

Figure 4.2: The entries in all bags after step 2 of `DPTH`.

Reasonable enumeration (step 3)

	C	E	F	B
Case 1 w = 0	1 0 1 1 1 0	0 1 0 0 1 0	1 0 1 0 1 1	1 0 1 0 0 1
Case 2 w = 0	1 0 1 1 1 0	0 1 0 0 1 0	1 0 1 0 1 1	0 1 1 0 0 1
Case 3 w = 0	1 0 1 1 1 0	0 1 0 0 1 0	1 0 1 0 1 1	0 1 0 1 0 1
Case 4 w = 0	0 1 1 1 1 0	1 0 0 0 1 0	0 1 1 0 1 1	1 0 1 0 0 1
Case 5 w = 0	0 1 1 1 1 0	1 0 0 0 1 0	0 1 1 0 1 1	0 1 1 0 0 1
Case 6 w = 0	0 1 1 1 1 0	1 0 0 0 1 0	0 1 1 0 1 1	0 1 0 1 0 1

	A	B	C
Case 1 w = 0	0 1 1 1 1 1	1 0 1 0 0 1	0 1 1 1 1 0
Case 2 w = 0	0 1 1 1 1 1	0 1 1 0 0 1	1 0 1 1 1 0

	B	D	E
Case 1 w = 0	0 1 0 1 0 1	1 1 0 0 1 1	1 0 0 0 1 0

Figure 4.3: The entries in all bags after step 3 of `DPTH`.

the size of the assignment set H in bag $\{C, E, F, B\}$ will be increased to 18. In the fifth step, the score of each assignment is further increased according to the recursive scoring function (Figure 4.5). And finally, in the last step, the optimal assignment is fetched (Figure 4.6), in which individual C inherits paternal origin copy $\{1, 1, 0\}$ from individual B with one recombination.

4.2 Discussion

`DPTH` captures the optimal assignment by exhaustively enumerating the potential assignments within each bag. As shown in this example, the optimal assignment is augmented into set H in step 3. The other 12 non-optimal assignments are redundant. `DPTH` will become less expensive, both in time and space, if it can effectively curb the growth of enumeration set H .

When designing an algorithm for an NP-hard problem, it is unavoidable to have a superpolynomial component if $P \neq NP$. However, efforts to limit exponential parts are supported by parameterized algorithm design theory [9]. Some hints about the future work will be given in Section 6.2.

The expansion of H set of piece {C, E, F, B} (step 2 – step 4)

	C	E	F	B		C	E	F	B
Case 1	1 0	0 1	1 0	1 0	Case 10	0 1	0 1	1 0	1 0
w = 0	1 1	0 0	1 0	1 0	w = 1	1 1	0 0	1 0	1 0
♦♣	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 2	0 1	1 0	0 1	1 0	Case 11	1 0	1 0	0 1	1 0
w = 0	1 1	0 0	1 0	1 0	w = 1	1 1	0 0	1 0	1 0
♦♣	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 3	1 0	0 1	1 0	0 1	Case 12	0 1	0 1	1 0	0 1
w = 0	1 1	0 0	1 0	1 0	w = 1	1 1	0 0	1 0	1 0
♣	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 4	0 1	1 0	0 1	0 1	Case 13	1 0	1 0	0 1	0 1
w = 0	1 1	0 0	1 0	1 0	w = 1	1 1	0 0	1 0	1 0
♣	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 5	1 0	0 1	1 0	0 1	Case 14	0 1	0 1	1 0	0 1
w = 0	1 1	0 0	1 0	0 1	w = 1	1 1	0 0	1 0	1 0
♣	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 6	0 1	1 0	0 1	0 1	Case 15	1 0	1 0	0 1	0 1
w = 0	1 1	0 0	1 0	0 1	w = 1	1 1	0 0	1 0	0 1
♣ ♥	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 7	1 0	1 0	1 0	1 0	Case 16	0 1	1 0	1 0	1 0
w = 1	1 1	0 0	1 0	1 0	w = 2	1 1	0 0	1 0	1 0
♠	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 8	1 0	1 0	1 0	0 1	Case 17	0 1	1 0	1 0	0 1
w = 1	1 1	0 0	1 0	1 0	w = 2	1 1	0 0	1 0	1 0
♠	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 9	1 0	1 0	1 0	0 1	Case 18	0 1	1 0	1 0	0 1
w = 1	1 1	0 0	1 0	0 1	w = 2	1 1	0 0	1 0	1 0
♠	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1

♦ marks entries added in step 2; ♠ marks entries added/modified in step 4;
♣ marks entries added/modified in step 3; ♥ marks optimal entries in step 5.

Figure 4.4: The expansion of enumeration set H from step 2 to step 4 of DPTH.

Evaluating the entries in H set of piece {C, E, F, B} (step 5)

	C	E	F	B		C	E	F	B
Case 1	1 0	0 1	1 0	1 0	Case 10	0 1	0 1	1 0	1 0
w = 3	1 1	0 0	1 0	1 0	w = 3	1 1	0 0	1 0	1 0
♦♣	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 2	0 1	1 0	0 1	1 0	Case 11	1 0	1 0	0 1	1 0
w = 1	1 1	0 0	1 0	1 0	w = 3	1 1	0 0	1 0	1 0
♦♣	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 3	1 0	0 1	1 0	0 1	Case 12	0 1	0 1	1 0	0 1
w = 3	1 1	0 0	1 0	1 0	w = 5	1 1	0 0	1 0	1 0
♣	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 4	0 1	1 0	0 1	0 1	Case 13	1 0	1 0	0 1	0 1
w = 3	1 1	0 0	1 0	1 0	w = 3	1 1	0 0	1 0	1 0
♣	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 5	1 0	0 1	1 0	0 1	Case 14	0 1	0 1	1 0	0 1
w = 3	1 1	0 0	1 0	0 1	w = 5	1 1	0 0	1 0	1 0
♣	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 6	0 1	1 0	0 1	0 1	Case 15	1 0	1 0	0 1	0 1
w = 1	1 1	0 0	1 0	0 1	w = 3	1 1	0 0	1 0	0 1
♣ ♥	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 7	1 0	1 0	1 0	1 0	Case 16	0 1	1 0	1 0	1 0
w = 3	1 1	0 0	1 0	1 0	w = 3	1 1	0 0	1 0	1 0
♠	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 8	1 0	1 0	1 0	0 1	Case 17	0 1	1 0	1 0	0 1
w = 3	1 1	0 0	1 0	1 0	w = 5	1 1	0 0	1 0	1 0
♠	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1
Case 9	1 0	1 0	1 0	0 1	Case 18	0 1	1 0	1 0	0 1
w = 3	1 1	0 0	1 0	0 1	w = 5	1 1	0 0	1 0	1 0
♠	1 0	1 0	1 1	0 1	♠	1 0	1 0	1 1	0 1

♦ marks entries added in step 2; ♠ marks entries added/modified in step 4;
♣ marks entries added/modified in step 3; ♥ marks optimal entries in step 5.

Figure 4.5: The quality of entries at root bag $\{C, E, F, B\}$ after step 5 of DPTH.

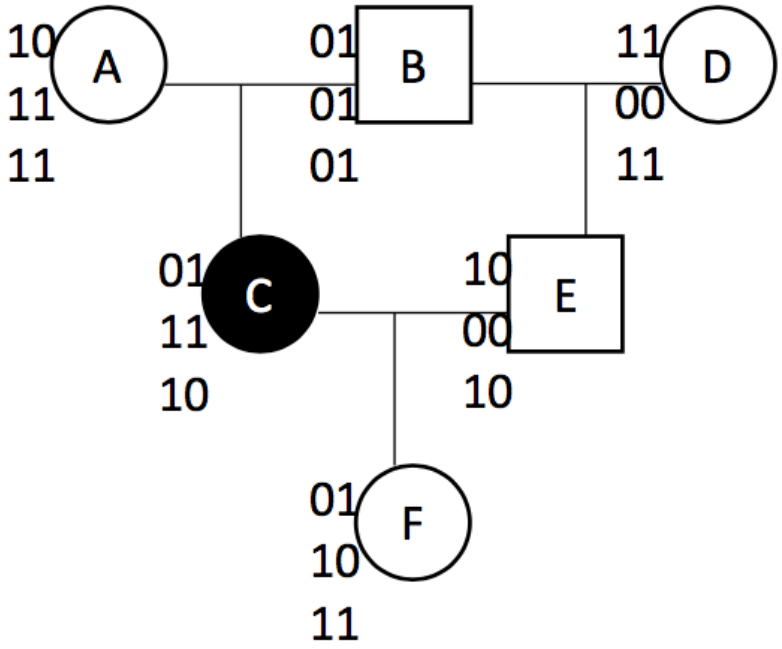


Figure 4.6: The optimal configuration of the pedigree.

Chapter 5

Extension

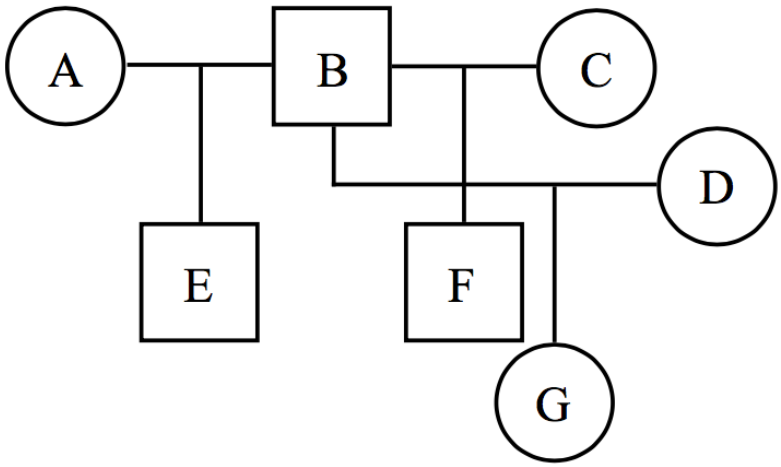
DPTH is based upon the idea of reasonable enumeration, where a homozygous locus predetermines the haplotypes of others in the family trio, in the absence of recombination. So, any haplotypes of individuals in the enumeration set should be triggered by at least one reasonable enumeration of a family trio. But, there exists a special case, where the optimal haplotype configuration requires that some individual has a recombination in each of their relationships.

In this chapter, an extension of DPTH, namely DPTH-Ex, will be proposed. DPTH-Ex will greatly expand the search space of DPTH.

An example of this special case is given in Figure 5.1. The reasonable enumeration of family trio $\{B, A, E\}$ will set the configuration of individual B to be $h'_B = \{\langle 1000111\rangle, \langle 0111000\rangle\}$; the reasonable enumeration of family trio $\{B, C, F\}$ will set the configuration of individual B to be $h''_B =$

$\{\langle 1100011\rangle, \langle 0011100\rangle\}$; the reasonable enumeration of family trio $\{B, D, G\}$ will set the configuration of individual B to be $h'''_B = \{\langle 1110001\rangle, \langle 0001110\rangle\}$. We put the number of recombinations needed to transform one haplotype to another as the distance between copies. In this case, the two by two distance of h'_B, h''_B, and h'''_B is four. So `DPTH` can only reach an assignment with $4+4 = 8$ recombination events. But there is a better choice for individual B, $h^{opt}_B = \{\langle 1111111\rangle, \langle 0000000\rangle\}$, where the relationship between these assignments is shown in Figure 5.2. Since h^{opt}_B can reach each of h'_B, h''_B, and h'''_B with distance 2, the overall cost of an assignment with h^{opt}_B can be brought down to $2 + 2 + 2 = 6$. This is a very rare case in real data, due to individual $\{B\}$ having recombination events in all three interactions. But `DPTH` should be extended to take this extreme case into consideration.

Continuing from Figure 3.4(c), Figure 5.3 illustrates the extra cases (such as f in Figure 5.3) that `DPTH-Ex` searches in addition to the reasonable enumeration based search of `DPTH`. If the cost of f ($weight(f) = distance(f, d) + distance(f, e) + weight(d) + weight(e)$) is lower than a, b, or any other entries in the configuration space of the root bag (Bag 1), then f can be the overall optimal assignment of the pedigree. Note that this optimal configuration, which is out of reach of all reasonable enumeration, can not be inferred by `DPTH`. Only `DPTH-Ex` can make that happen.



(a)

Genotype:

A	B	C	D	E	F	G
1	2	1	1	2	2	2
0	2	1	1	2	2	2
0	2	0	1	2	2	2
0	2	0	0	2	2	2
1	2	0	0	2	2	2
1	2	1	0	2	2	2
1	2	1	1	2	2	2

(b)

Figure 5.1: An exception pedigree for which `DPTH` can not achieve optimal configuration.

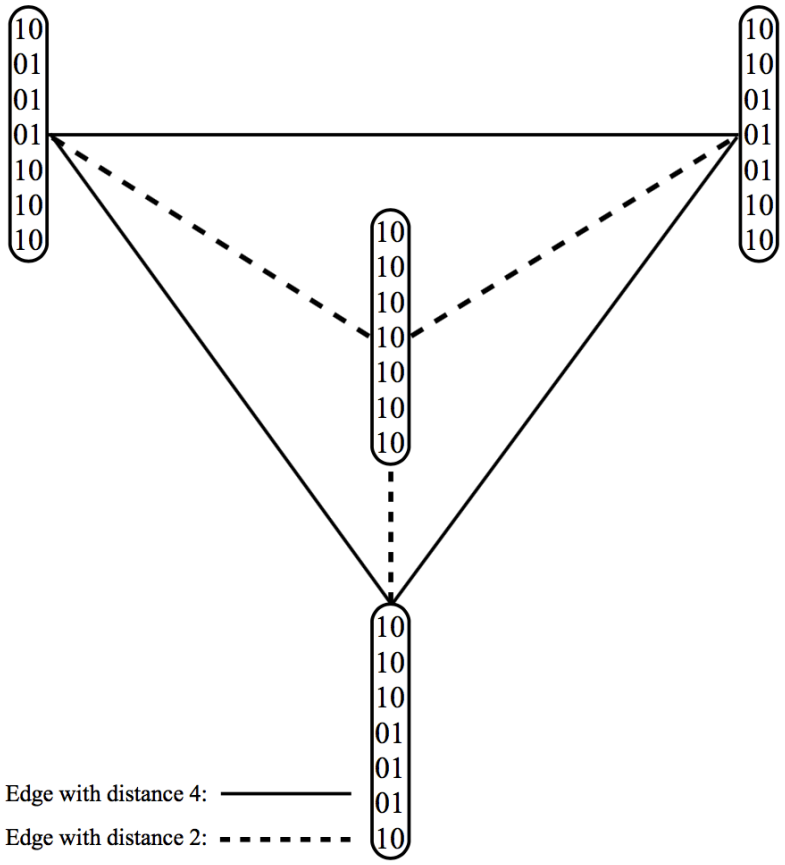


Figure 5.2: Configuration (at the center) that can not be detected by reasonable enumeration.

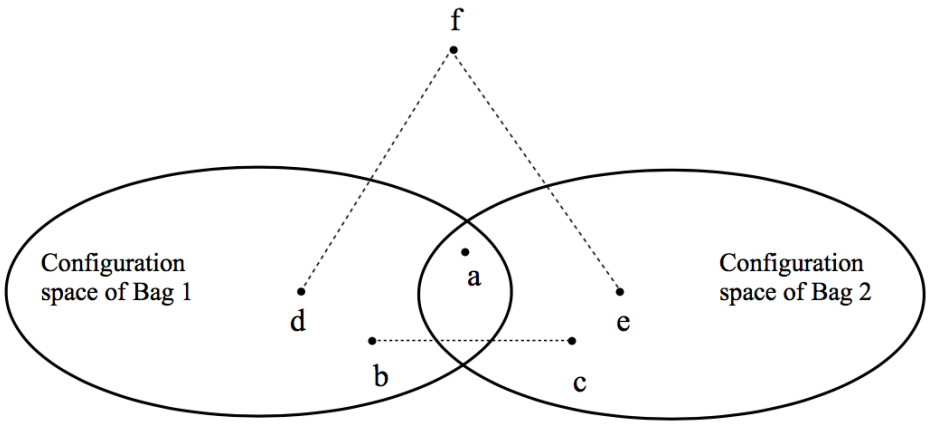


Figure 5.3: A visualized high-level introduction to the methodology of DPTH-Ex.

5.1 Methodology of DPTH-Ex

To put the necessary non-reasonable-enumerable haplotype assignment into the enumeration process, step 3 and step 4 of DPTH need to be modified to put the extra copies in.

Step 3 of DPTH-Ex will augment all possible haplotypes of individuals to augment on, instead of augmenting what exists in the substructures. The pseudocode of step 3 of DPTH-Ex is shown in Algorithm 6.

Step 4 of DPTH-Ex will also augment all possible configurations of individuals to augment on, rather than augmenting what exists below. The pseudocode of step 4 of DPTH-Ex is shown in Algorithm 7.

Algorithm 6: InitForiIndi[Extension]

Input : tree decomposition td

Output: updated copy of each trio member and value r

```
1  begin
2      postOrderTrav ← PostOrderTraversal(td)
3      foreach bag p_i in postOrderTrav do
4          children ← substructures rooted at p_i
5          if children.size = 0 then
6              continue
7          foreach foreign individual indi_j in bag p_i do
8              allCopy ← all possible haplotypes of indi_j
9              Augment the enumeration set of p_i with allCopy of indi_j
10     return
```

Algorithm 7: RecursiveLabelMultiCopyTrio[Extension]

Input : tree decomposition td, the root bag r of td

Output: updated copy of each trio member and value r

```
1  begin
2      if r is the leaf in td then
3          return
4      else
5          children ← substructures rooted at r
6          foreach bag b_i of children do
7              RecursiveLabelMultiCopyTrio[Extension](b_i)
8          foreach bag p_i of children do
9              copyset ← {}
10             if r.father resolved in p_i then
11                 copyset ← get all possible copies of r.father
12             if r.mother resolved in p_i then
13                 copyset ← get all possible copies of r.mother
14             if r.child resolved in p_i then
15                 copyset ← get all possible copies of r.child
16             Augment the enumeration set H of r with copyset
17     return
```

5.2 Time complexity of DPTH-Ex

DPTH-Ex will augment $O(2^{\eta-1})$ copies, where η is the number of heterozygous loci of individuals to augment, into the enumeration set. This augmentation process is apparently more expensive than the augmentation process of DPTH. But the time complexity analysis for DPTH put $O(2^{m-1})$ as the upper bound of the extra configuration set to be augmented on, where m is the number of loci. The value of m is greater or equal to η, so that "Big O" time complexity of DPTH given should also be the time analysis for DPTH-Ex: $O(mn^2 2^{2mw+4m})$.

5.3 Correctness proof of DPTH-Ex

DPTH is a practical technique to solve the MRHC problem under the assumption that recombination events are expected to be few. DPTH-Ex will do an extensive search over the tree decomposition, and we will prove its completeness in Theorem 5.3.1.

Theorem 5.3.1. *The enumeration set H of* DPTH-Ex *is optimal configuration complete (Correctness proof).*

Proof. If the pedigree has only one family trio, then the corresponding tree decomposition has only one bag, and the optimal assignment can be reached by reasonable enumeration.

If the pedigree has more than one family trio, then the inference process will perform a bottom-up search in the tree decomposition of the family pedigree.

For each leaf bag, which is equal to the pedigree with one family trio, the optimal bag-wide assignment can be found in the reasonable enumeration.
Each non-leaf bag searches for an optimal assignment based on its reasonable enumeration. The search process is based upon augmenting all possible copies (according to the genotype sequence) of individuals that are also in the substructure. The augmenting process is simply the Cartesian product of the old set and the copies to augment. Any assignment that is out of the scope will only violate reasonable enumeration and increase the recombination number, but not better accommodate assignment of bags below. So the exhaustive enumeration process is complete. □

5.4 Discussion

The extension from `DPTH` to `DPTH-Ex` is to cover some very rare pedigrees for which the optimal assignment can not be reasonably enumerated.
Such special cases will make the optimal assignment be outside the search space of `DPTH` based on reasonable enumeration. But this is a rare case, for the recombination event is rare, and it will be even rarer to occur multiple times in a single individual.
We think that in practice `DPTH` is sufficient, but `DPTH-Ex` should be used when the result of `DPTH` is not reasonable. If `DPTH` yields an optimal assignment that has some "expensive" edges, then a better alternative configuration, with less "expensive" edges, may exist beyond the search space of `DPTH`.

Chapter 6

Conclusion

The increasing availability of genetic data is pushing the human genetic investigation from theory motivated to data motivated. MRHC is one such problem, designated to make inferences from genotype sequences on a pedigree.

The algorithm that can solve the MRHC problem can reduce the cost of genetic data acquisition. Also, accurate haplotype configurations on a pedigree could pave the road for more detailed gene-disease association studies.

Our method is an "expert" of enumeration. By introducing the structure of tree decomposition, we decompose the pedigree into pieces. Within these pieces, the enumeration of all possible configurations is affordable. We believe such a search technique can be applied to a variety of other problems.

6.1 Summary of the methods

We formulate a parameterized $O(mn^2 2^{2mw+4m})$ time algorithm for the MRHC problem using dynamic programming over a tree decomposition of the pedigree. The method first enumerates all potential optimal configurations of each subproblem following the structure of the tree decomposition. It then finalizes the optimal assignment selection using recursive calls with dynamic programming.

Doan's result [6] unveiled a $O(2^k n^2)$ parameterized time algorithm over a general pedigree with individuals with 2 loci, where k is the number of recombination events. `DPTH` and `DPTH-Ex`, on the other hand, have 2 parameters, the number of loci m and treewidth w. For $m = 2$, `DPTH` has running time of $O(n^2 2^{4w+8})$, independent of the number of recombinations k. This algorithm thus provides faster computation for near-tree pedigrees even when there can be many recombinations.

6.2 Future work

Future work may concentrate on two aspects. Firstly, the algorithm may be refined, such as potentially merging multiple steps to improve the efficiency of each recursion. Secondly, reducing the maximum number of substructures in the tree decomposition may be applied to strategically reduce the growth of H during the exhaustive enumeration process.

Optimizing the parameter w is desirable, and may be possible by using other

techniques for computing tree decompositions that can handle common pedigree structures, but minimizing it in a complex pedigree is NP-hard in general [9].

Bibliography

[1] Hans-Joachim Böckenhauer and Dirk Bongartz, *Algorithmic aspects of bioinformatics*, Springer Berlin Heidelberg, 2007.

[2] Ali H Brivanlou and James E Darnell, *Signal transduction and the control of gene expression*, Science **295** (2002), no. 5556, 813–818.

[3] Mee Yee Chan, Wun-Tat Chan, et al., *Linear-time haplotype inference on pedigrees without recombinations and mating loops*, SIAM J. Comput. **38** (2009), no. 6, 2179–2197.

[4] Roger Cox, Nourdine Bouzekri, and Sabrina et al. Martin, *Angiotensin-1-converting enzyme (ace) plasma concentration is influenced by multiple ace-linked quantitative trait nucleotides*, Human Molecular Genetics **11** (2002), no. 23, 2969.

[5] Duong D. Doan and Patricia A. Evans, *An fpt haplotyping algorithm on pedigrees with a small number of sites*, Algorithms for Molecular Biology **6** (2011), no. 1, 8.

[6] Duong D Doan and Patricia A Evans, *Haplotype inference in general pedigrees with two sites*, BMC proceedings, vol. 5, BioMed Central, 2011, p. S6.

[7] Duong Dai Doan and Patricia A. Evans, *Fixed-parameter algorithm for haplotype inferences on general pedigrees with small number of sites*, Algorithms in Bioinformatics, 10th International Workshop, WABI, LNCS, vol. 6293, Springer, 2010, pp. 124–135.

[8] Duong Dai Doan, Patricia A. Evans, and Joseph D. Horton, *A near-linear time algorithm for haplotype determination on general pedigrees*, J. Comp. Biol. **17** (2010), no. 10, 1451–1465.

[9] Rodney G. Downey and Michael R. Fellows, *Fundamentals of parameterized complexity*, Texts in Computer Science, Springer, 2013.

[10] Jonathan L. Haines, *Chromlook: An interactive program for error detection and mapping in reference linkage data*, Genomics **14** (1992), no. 2, 517 – 519.

[11] A. D. Hershey and Martha Chase, *Independent functions of viral protein and nucleic acid in growth of bacteriophage*, The Journal of General Physiology **36** (1952), no. 1, 39–56.

[12] Ting Hu, Nicholas A. Sinnott-Armstrong, Jeff W. Kiralis, Angeline S. Andrew, Margaret R. Karagas, and Jason H. Moore, *Characterizing ge-*

netic interactions in human disease association studies using statistical epistasis networks, BMC Bioinformatics **12** (2011), no. 1, 364.

[13] August Y Huang, Xiaojing Xu, Y Ye Adam, Qixi Wu, Linlin Yan, Boxun Zhao, Xiaoxu Yang, Yao He, Sheng Wang, Zheng Zhang, et al., *Postzygotic single-nucleotide mosaicisms in whole-genome sequences of clinically unremarkable individuals*, Cell research **24** (2014), no. 11, 1311–1327.

[14] Bonnie Kirkpatrick, *Haplotype inference for pedigrees with few recombinations*, Bioinformatics Research and Applications - 12th International Symposium, ISBRA 2016, LNCS, vol. 9683, Springer, 2016, pp. 269–283.

[15] Bonnie Kirkpatrick, Eran Halperin, and Richard M. Karp, *Haplotype inference in complex pedigrees*, Journal of Computational Biology **17** (2010), no. 3, 269–280.

[16] Jocelyn E Krebs, Benjamin Lewin, Elliott S Goldstein, and Stephen T Kilpatrick, *Lewin's genes xi*, Jones & Bartlett Publishers, 2014.

[17] En-Yu Lai, Wei-Bung Wang, Tao Jiang, and Kun-Pin Wu, *A linear-time algorithm for reconstructing zero-recombinant haplotype configuration on a pedigree*, BMC Bioinformatics **13** (2012), no. S-17, S19.

[18] Jing Li and Tao Jiang, *Efficient inference of haplotypes from genotypes on a pedigree*, J. Bioinformatics and Computational Biology **1** (2003), no. 1, 41–70.

[19] Jing Li and Tao Jiang, *Efficient rule-based haplotyping algorithms for pedigree data*, Proceedings of the seventh annual international conference on Research in computational molecular biology, ACM, 2003, pp. 197–206.

[20] X. Li and J. Li, *Efficient haplotype inference from pedigrees with missing data using linear systems with disjoint-set data structure*, Computational Systems Bioinformatics **7** (2008), 297–308.

[21] Lan Liu and Tao Jiang, *A linear-time algorithm for reconstructing zero-recombinant haplotype configuration on pedigrees without mating loops*, J. Comb. Optim. **19** (2010), no. 2, 217–240.

[22] Thomas M. Mitchell, *Machine learning*, 1 ed., McGraw-Hill, Inc., New York, NY, USA, 1997.

[23] Ardeshir Nejati-Javaremi and Charles Smith, *Assigning linkage haplotypes from parent and progeny genotypes*, Genetics **142** (1996), no. 4, 1363–1367.

[24] D. Qian and L. Beckmann, *Minimum-recombinant haplotyping in pedigrees*, The American Journal of Human Genetics **70** (2002).

[25] Pradip Tapadar, Saurabh Ghosh, and Partha P Majumder, *Haplotyping in pedigrees via a genetic algorithm*, Human heredity **50** (2000), no. 1, 43–56.

[26] James D Watson, Francis HC Crick, et al., *Molecular structure of nucleic acids*, Nature **171** (1953), no. 4356, 737–738.

[27] Edmund B Wilson, *Cell in development and heredity, 3rd. rev*, Macmillan Company.; New York, 1925.

[28] Jing Xiao, Lan Liu, Lirong Xia, and Tao Jiang, *Fast elimination of redundant linear equations and reconstruction of recombination-free mendelian inheritance on a pedigree*, Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007 (Nikhil Bansal, Kirk Pruhs, and Clifford Stein, eds.), SIAM, 2007, pp. 655–664.

[29] Jing Xiao, Lan Liu, Lirong Xia, and Tao Jiang, *Efficient algorithms for reconstructing zero-recombinant haplotypes on a pedigree based on fast elimination of redundant linear equations*, SIAM Journal on Computing **38** (2009), no. 6, 2198–2219.

[30] Jing Xiao, Tiancheng Lou, and Tao Jiang, *An efficient algorithm for haplotype inference on pedigrees with a small number of recombinants*, Algorithmica **62** (2012), no. 3-4, 951–981.

[31] Xiaojing Xu et al., *Amplicon resequencing identified parental mosaicism for approximately 10% of de novo scn1a mutations in children with dravet syndrome*, Human Mutation **36** (2015), no. 9, 861–872.

[32] Kui Zhang, Fengzhu Sun, and Hongyu Zhao, *Haplore: a program for haplotype reconstruction in general pedigrees without recombination*, Bioinformatics **21** (2005), no. 1, 90.

Vita

Candidate's full name: Zhendong Sha
University attended:

September 2017 - present, Memorial University of Newfoundland, Canada,
PhD Candidate, Computer Science

September 2014 - December 2018, University of New Brunswick, Canada,
Master, Computer Science

September 2013 - present, Southeast University, China,
Master, Software Engineering

September 2011 - July 2013, Nanjing University Jingling College, China,
Bachelor, Electronic Information Science and technology

September 2009 - July 2011, Nanjing Institute of Railway Technology, China