Using tree decomposition for general pedigree inference

Zhendong Sha & Patricia A. Evans
University of New Brunswick - Faculty of Computer Science

Faculty of Computer Science
UNB FREDERICTON

Introduction

Genetic 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; however, researchers often initially gather **genotype sequences** over a finite alphabet $\{0,1,2\}$, where 2 indicates difference, rather than a pair of sequence over a finite alphabet $\{0,1\}$, namely **haplotype sequences**. This is partly because of economic and time concerns. As a result, an effective computational method of haplotype phasing, to infer haplotype from genotypes, is needed. In practice, the hidden **recombination event** and the mating loop complicate the phasing process.

The problem

Given a family pedigree with each individual marked with genotype, can a pair of haplotypes for all individuals be inferred that minimizes the number of recombinations between haplotypes? We propose the **DPTH** algorithm to find an optimal solution to the MRHC problem.

Haplotype inference

	homozygous site		heterozygous site	
haplotype of diploid genome	0	1	0	1
	0	1	1	0
genotype of diploid genome	0	1	2	2

inference

The strategy

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 treelike, and having a tree decomposition over the pedigree can greatly reduce the search space.
A tree decomposition 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 $\cup_{x\in\tau} T_x = V$ with 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 $T_x \cap T_z \subseteq T_y$.

Method

DPTH first reorganizes the pedigree using tree decomposition, and then enumerates the potential optimal haplotype settings within each bag. For each enumerated assignment, we define a score to indicate the quality (number of recombinations), from which the optimal assignment will be extracted.

Local Perspective

Assuming that there is no recombination event in the **parent-offspring trio**, then we can enumerate all **reasonable haplotype assignments** (0 recombination).

Global Perspective

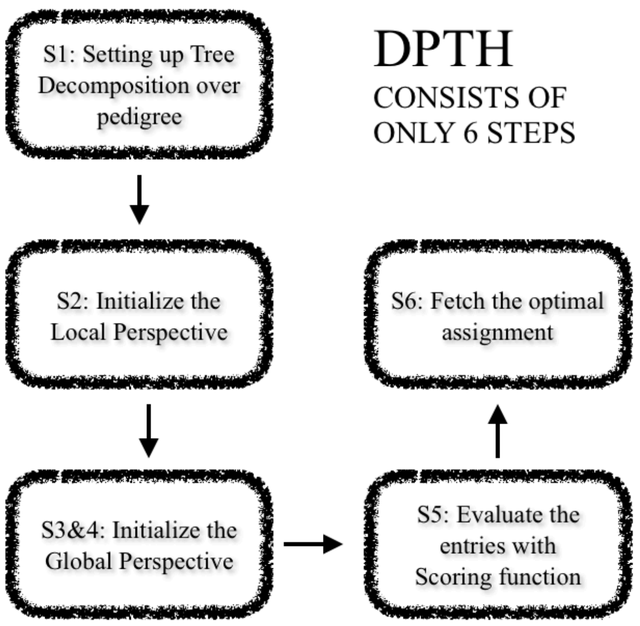
Some connections(DPTH's Global Perspective) have to be set up in order to synchronize the assignment set in each tree organized bags. Cartesian product will be applied to combine the features from sub-structures.

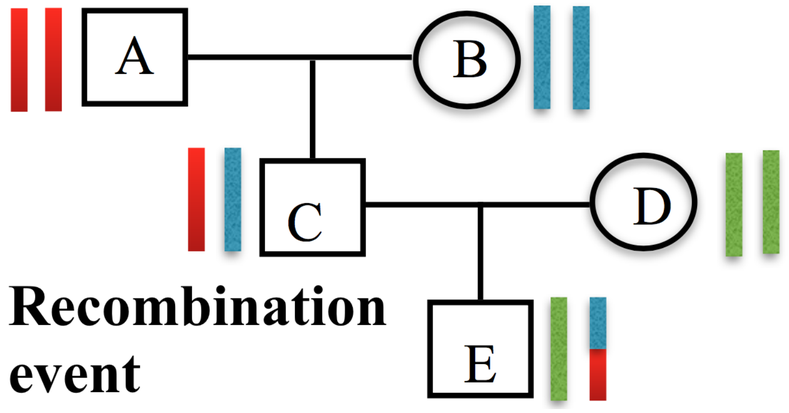
Scoring Function

DPTH defines a recursive scoring function to evaluate the previous enumerated entries:

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

where h_R and h_j each represents a enumerated entry, *Children* is the set of sub-structure, and the function d measures the distance between 2 configurations.





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.